APLICACIÓN DE TÉCNICAS DE CLUSTERING PARA LA ESTIMACIÓN DEL ESFUERZO EN LA CONSTRUCCIÓN DE PROYECTOS SOFTWARE

TESIS DOCTORAL

Alcalá de Henares, Mayo 2006

Autor:

Miguel Garre Rubio

Ingeniero en Informática

Directores:

Dr. Juan José Cuadrado Gallego

Dr. Miguel Ángel Sicilia Urbán

Esta tesis se la dedico a mis padres, que me dieron la vida, y en especial a mi padre, José Garre, que descanse en paz.

Una vez concluido el trabajo de tesis doctoral titulado: **"Aplicación de técnicas de clustering para la estimación del esfuerzo en la construcción de proyectos software"**, realizado por D. Miguel Garre Rubio y del que somos directores, estimamos que dicho trabajo tiene suficientes méritos teóricos, que se han contrastado adecuadamente mediante validaciones experimentales y son altamente novedosos.

Por ello consideramos que procede su defensa pública.

Y para que así conste, firmamos la presente en Alcalá de Henares, a 15 de Mayo de dos mil seis.

Atentamente,

Los directores de la tesis:

Dr. Juan José Cuadrado Gallego

Dr. Miguel Ángel Sicilia Urbán

Dña. ENRIQUETA MUEL MUEL, Profesora Titular de Universidad del Área de Ciencias de la Computación e Inteligencia Artificial, en calidad de Directora del Departamento de Ciencias de la Computación

CERTIFICO: Que la Tesis Doctoral titulada **"Aplicación de técnicas de clustering para la estimación del esfuerzo en la construcción de proyectos software"**, realizada por D. Miguel Garre Rubio y dirigida por Dr. Juan José Cuadrado Gallego y Dr. Miguel Ángel Sicilia Urbán, reúne los requisitos para su presentación y defensa pública.

Y para que así conste, firmo la presente en Alcalá de Henares, a 15 de Mayo de dos mil seis.

Fdo: Dra. Enriqueta Muel Muel

APLICACIÓN DE TÉCNICAS DE CLUSTERING PARA LA ESTIMACIÓN DEL ESFUERZO EN LA CONSTRUCCIÓN DE PROYECTOS SOFTWARE

TESIS DOCTORAL

ÁREA: Lenguajes y Sistemas Informáticos

DEPARTAMENTO: Ciencias de la Computación

UNIVERSIDAD DE ALCALÁ

Alcalá de Henares, Mayo de 2006

Tabla de contenidos

Índice de tablas

Índice de figuras

Resumen

Los modelos paramétricos de estimación de coste del software basados en la utilización de una única ecuación que representa a toda una base de datos de proyectos, ofrecen resultados poco satisfactorios. Esto se debe a diversas razones, una de las cuales reside en la heterogeneidad de los proyectos utilizados en la obtención de dicha ecuación.

Se trata entonces de procurar salvar este inconveniente. Para conseguir esto, surge la idea de dividir esta base de datos de proyectos en grupos de ellos, de manera que los proyectos integrantes sean más homogéneos entre si. Para realizar esta división de una forma automática y de manera que, efectivamente, proyectos similares se encuentren en el mismo grupo, se recurre a técnicas de Inteligencia Artificial, en concreto a algoritmos de agrupamiento automático o *clustering*.

De ello surge un nuevo modelo, al que se le denomina modelo de estimación paramétrico segmentado. Consiste en la obtención de un conjunto de *clusters*, o grupos de proyectos, tras la aplicación de un algoritmo de *clustering* sobre la totalidad de la base de datos de proyectos. Los clusters así obtenidos ofrecen unas características más homogéneas que las que presentaban por separado los proyectos que los constituyen.

Para cada uno de los grupos de proyectos obtenidos de esta manera se obtendrá, mediante análisis de regresión, su propia ecuación paramétrica. Estas ecuaciones ofrecen, en término medio, después de las pruebas realizadas, mejores resultados que el uso exclusivo de una única ecuación para todos los proyectos de la base de datos.

Este proceso se puede repetir de forma recursiva, si se considera conveniente, de

manera que sucesivamente se proceda al refinamiento de los clusters que se van construyendo.

Dividir directamente los proyectos de la base de datos, no aprovecha el conocimiento previo que se pueda tener por parte de expertos sobre el comportamiento de los mismos. Con el fin de analizar este aspecto, se han realizado diferentes experimentos, que antes de llevar a cabo el proceso de *clusterización*, realizan un particionado previo de los proyectos, utilizando el conocimiento que personas expertas tienen sobre la influencia que tienen ciertos conductores de coste sobre el esfuerzo que requiere la realización de un proyecto. De esta manera, a la hora de aplicar el algoritmo de agrupamiento, se partirá de la experiencia y conocimiento ofrecido por estos expertos, que se utilizará para realizar un particionado previo de los datos.

La forma en la que se ha llevado a cabo la evaluación del nuevo modelo paramétrico segmentado ha consistido en la comparación con el modelo paramétrico clásico, mediante la utilización de los índices MMRE (*Mean Magnitude of Relative Error*) y PRED(l [1]) (*Prediction level*). Estos índices han mostrado unos resultados más satisfactorios utilizando el modelo aquí expuesto que utilizando el tradicional.

Palabras clave: clustering, segmentación, segmentación recursiva, algoritmo EM, estimación del coste del software, modelo de estimación paramétrico segmentado, ingeniería del software

[1]Según Conte et. al (Conte et al. 1986), un valor apropiado para l podría ser 0,25 o 0,3.

Abstract

Parametric software estimation models rely on the availability of historical project databases from which estimation models are derived. In this case, a single mathematical model cannot properly capture the diverse nature of the projects under consideration. The use of a single mathematical model offers poor quality of adjustment, due to several factors, one of which consists on heterogeneity of data used to elaborate the model.

It is necessary to deal with this problem. The idea of splitting the project database in projects groups is the main motivation of this work. The members of these groups show a more homogeneous relationship between them. This task is automatically done using Artificial Intelligence techniques such us clustering algorithms, that divides data into segments of related projets.

A new estimation model is presented in this work, called the segmented parametric software estimation model, which produces a set of clusters made up each of them by many different projects, after the use of a clustering algorithm over the entire project database. The projects clusters obtained in this manner present more homogeneous characteristics than others not clusterized.

A mathematical model is given for each cluster get in this manner. This mathematical model consist of a parametric equation obtained by means of regression analysis. The quality of adjustment of this multi model is better than the single parametric model in the evaluations carried out.

The task can be carried out recursively if considered appropriate, getting even more

homogeneous subclusters in consecutive steps.

The straightforward clustering process over all the projects does not use previous expert knowledge that they could get over them. Using this principle a new proposal has been made, this consists on creating a partition of projets before the clustering process can be done. This partition process divide projects, using expert knowledge, in groups of similar characteristics based on the influence that some cost drivers have over the effort estimation. In this manner, the clustering process will be improved due to a more suitable framework.

The MMRE (*Mean Magnitude of Relative Error*) and PRED(l [2]) (*Prediction level*) measurements have been used to compare the two models, the segmented parametric model versus the not segmented parametric, to evaluate the accuracy and quality of adjustment of the technique proposed in this work. The segmented parametric software estimation model has provided better results than the not segmented model.

Keywords: clustering, segmentation, recursive segmentation, EM algorithm, software effort estimation, segmented parametric software estimation model, Software Engineering

[2] *Conte et. al (Conte et al. 1986) establish* $l = 0,25$ *or* $l = 0,3$ *as good values.*

Agradecimientos

Tengo que agradecerle el poder haber realizado esta tesis a varias personas, y tengo que empezar por supuesto por mi esposa y por mi hijo, los cuales me han animado a seguir adelante con su sola presencia a mi lado, perdonándome el tiempo que no les he podido dedicar y que me gustaría haberles dedicado. A ti Trini te doy las gracias por todo lo que has aguantado a mi lado, y que seguramente te quedará por aguantar, me has apoyado siempre en mi trabajo y me has empujado en muchas ocasiones en las que necesitaba salir de algún bache. Y a ti José Miguel, darte las gracias por tu sonrisa.

Nada de este trabajo se habría realizado si un buen día la Dra. Dña. Elena García Barriocanal no se hubiese interesado por mí, presentándome a mis dos grandes maestros y co-directores de esta tesis, se trata del Dr. D. Juan José Cuadrado Gallego y el Dr. D. Miguel Ángel Sicilia Urbán. Mi agradecimiento pues a los tres, ya que han conseguido sacar de mí lo que hacía tiempo estaba aguardando el momento de surgir, solamente los necesitaba a ellos, solamente, y ahí estaban, están. Gracias una vez más a vosotros, mis bienhechores, ya que durante el tiempo que he estado trabajando con vosotros, y espero seguir haciendo, he aprendido bastante, enriqueciéndome tanto en el terreno profesional como en el personal.

También tengo que agradecerte a ti compañero y amigo, que has sufrido junto conmigo todo el trabajo realizado, en el cual has participado muy activamente, todo lo que has hecho por mí de forma totalmente desinteresada. Para formar parte de nuestra profesión hay que estar hecho de una madera especial y esa es la tuya Dr. D. Mario Charro Cubero. Gracias.

Por último y para que nadie se quede fuera de mis reconocimientos, agradecer a todas las personas que, desde el anonimato y también de forma desinteresada, también me han ayudado en la realización de esta tesis.

Gracias a todos por vuestra ayuda, tiempo y dedicación.

Alcalá de Henares, 15 de mayo de 2006

Miguel Garre Rubio

Capítulo 1

Introducción

Poor management can increase software costs more rapidly than any other factor.

Barry Boehm

El objetivo de este capítulo es ofrecer una síntesis del problema que se pretende resolver, describiendo tanto los objetivos como el método de trabajo utilizado en la investigación. Se resumen las principales aportaciones, que se irán posteriormente detallando a lo largo del documento.

1.1. Planteamiento del problema

Según el informe CHAOS realizado en 1994 por The Standish Group[1], en Estados Unidos se realizaban al año aproximadamente 175.000 proyectos de desarrollo software, con un coste total de 250 billones de dólares. El 31,1 % de los mismos se cancelaba antes de que se completara. Solamente el 16,2 % de los proyectos se solía acabar en el tiempo estimado. Además el 52,7 % de proyectos, que se acababan, costaban al final de su realización un 189 % más sobre el coste económico estimado al principio de su desarrollo.

Basándose en esta investigación, The Standish Group estimó que en el año 1995, las principales compañías americanas y agencias del gobierno, gastarían 81 billones de dólares, debido a proyectos de software que se cancelarían. Además, estas mismas compañías gastarían 59 billones de dólares en proyectos que se acabarían, pero fuera de plazo.

[1] http://www.standishgroup.com.

Otro estudio más reciente ha sido el realizado por Holgeid (Holgeid et al. 2000), en el que se presentan los resultados obtenidos al realizar un test de 40 preguntas a 53 compañías noruegas, dedicadas al desarrollo y mantenimiento de software. En este artículo se realiza una comparativa con los resultados obtenidos en estudios anteriores, quedando reflejados los nuevos valores en cuanto al tiempo dedicado por las compañías al mantenimiento y desarrollo. Para mantenimiento se dedica un 41 %, para desarrollo un 17 % y un 42 % para el resto de actividades. Estos, junto con otros datos, vuelven a reflejar la importancia que tiene la construcción de proyectos de software, y el esfuerzo que es necesario realizar para que estas tareas se realicen de la forma más rápida y económica posible.

Las grandes pérdidas económicas ocasionadas a las empresas, se deben a diferentes razones, atribuibles a la propia organización, a los clientes, y otras entidades relacionadas con el sistema. La Ingeniería del Software ha tenido como objetivo desde sus inicios en los años 60, ayudar a las compañías a obtener software de calidad, que cumpla con los requisitos demandados por los usuarios, se termine en plazo sin sobrepasar el coste previsto para su realización. Para conseguir esto se han desarrollado y puesto a disposición de las compañías diversas metodologías y herramientas, que ayudan a realizar el proceso de desarrollo software de forma más eficiente y de manera que dé lugar a un producto con las características anteriormente mencionadas. Una de las áreas que ha sido objeto de investigación durante estos años, para este fin, ha sido la de la estimación del coste del software.

Como se verá en el próximo capítulo, existen numerosos modelos de estimación del coste del software llevados a cabo por Boehm (Boehm, Abts & Chulani 2000), los cuales tratan de dar una información estimada sobre el coste temporal, de personal, económico, etc., que podría necesitar la realización de un proyecto de software. Esta estimación, para que sea útil, debería realizarse preferentemente en la etapa de establecimiento de

requisitos software, ya que al comienzo del proyecto es cuando es más necesario conocer de forma aproximada cuanto tiempo se tardará en llevarlo a cabo, contando con un número especifico de personas y asumiendo un coste económico determinado.

Uno de estos modelos es el modelo paramétrico, que utiliza funciones matemáticas en las que la variable dependiente es el esfuerzo o el tiempo, y las variables independientes son diferentes aspectos del proyecto o del producto o de ambos. Estas ecuaciones son como por ejemplo las que utilizan la función potencia $e = A_0 \cdot s^{A_1}$, aunque también pueden aparecer en forma lineal $e = A_0 + A_1 \cdot s$, donde e es el esfuerzo estimado, A_0 y A_1 parámetros a determinar, y s alguna medida del tamaño del proyecto, tal como puntos de función.

Este modelo necesita, para su calibración, una base de datos de proyectos históricos, con los que obtiene los valores de los parámetros A_0 y A_1. Utilizar una única fórmula de este tipo para toda la base de datos de proyectos, donde los proyectos son heterogéneos, proporciona en muchos casos resultados poco satisfactorios. Por ejemplo, utilizando la herramienta Reality de la base de datos de proyectos ISBSG versión 7 (International Software Benchmarking Standard Group[2]), aplicada a 709 proyectos, se obtiene la siguiente fórmula:

$$e = 47,73 \cdot s^{0,76}$$

donde el esfuerzo se expresa en horas, y el tamaño en puntos de función. Un análisis de la bondad del ajuste ofrece como resultados MMRE=1,18 y PRED(0,3)=25,6 %[3]. Ambas medidas son difícilmente aceptables, dado el alto grado de desviación sobre la inmensa mayoría de los datos. Algunos autores tales como Abran y Reifer (Abran 2003, Reifer et al. 1999) han sugerido que la segmentación de los datos contenidos en las bases de datos históricas podría ser un camino adecuado para la obtención de ecuaciones matemáticas que proporcionasen una mayor exactitud en las estimaciones. Esta idea es la motivación original de esta tesis doctoral.

[2]http://www.isbsg.org/.
[3]Ambas medidas serán definidas en el capítulo 5.

En este trabajo se muestra la investigación realizada sobre el uso de diferentes algoritmos de *clustering*[4] (se verán en el capítulo siguiente), para realizar la segmentación de una base de datos de proyectos de software, y así poder obtener una función paramétrica ajustada específicamente para cada uno de estos segmentos[5]. Las técnicas de clusterización empleadas pertenecen al área de Inteligencia Artificial, en concreto Aprendizaje Automático (*Machine Learning*) (Mitchell 1997), y realizan un aprendizaje no supervisado, es decir, proceden a la segmentación de proyectos sin necesidad de que un supervisor indique el número de segmentos a obtener. El primer trabajo que aborda este tema ha sido el de J.J. Cuadrado (Cuadrado et al. 2004) en el que se aplica el algoritmo EM sobre la base de datos de proyectos históricos ISBSG versión 8.

1.2. Objetivos y aportaciones originales

El objetivo principal de este trabajo consiste en estudiar las propiedades de los modelos paramétricos segmentados para la estimación del coste del software en Ingeniería del Software. Es decir, analizar la bondad de las predicciones que se obtienen al utilizar varias funciones de estimación paramétricas, ajustadas a cada uno de los segmentos obtenidos al aplicar técnicas de clustering para llevar a cabo la segmentación de bases de datos de proyectos, así como realizar la comparación de estos resultados con los obtenidos mediante un modelo paramétrico que utiliza una única función.

Para alcanzar este objetivo general, se plantean los siguientes sub-objetivos:

(O1) Comparar las propiedades predictivas de los modelos paramétricos segmentados con modelos paramétricos no segmentados mediante el uso de datos empíricos

(O2) Diseñar un algoritmo genérico para la generación de modelos paramétricos segmentados de Ingeniería del Software

(O3) Estudiar la influencia de la utilización de conductores de coste (*cost drivers*[6])

[4]Este término no se traducirá al castellano, por su amplia aceptación en su forma inglesa en países de lengua hispana.

[5]Los términos segmento y cluster se utilizan de forma indistinta, con un significado idéntico, así como clusterización y segmentación.

[6]Este término no será traducido al castellano por su amplia aceptación anglo-sajona.

conocidos, en la generación de modelos paramétricos segmentados de Ingeniería del Software

(O4) Comparar los resultados obtenidos aplicando diferentes algoritmos de clustering en la generación de modelos paramétricos segmentados

La aportación fundamental de la tesis será la creación de un modelo de estimación del coste del software híbrido, que utiliza un modelo de estimación paramétrico junto a técnicas de clustering. Es decir, se obtendrá un modelo paramétrico segmentado.

Entre las aportaciones adicionales significativas que surgen como resultado del proceso de investigación, es posible destacar las siguientes:

- Construcción de una herramienta *Recursive Clustering Tool* (RCT) que, basándose en el algoritmo de clustering EM, obtiene las diferentes ecuaciones paramétricas asociadas a cada uno de los segmentos obtenidos al realizar la clusterización de proyectos, permitiendo realizar diferentes análisis sobre las mismas.

- Se ha confirmado la influencia de los cost drivers METHO (uso de una metodología) y CASET (uso de una herramienta CASE) sobre el proceso paramétrico segmentado de estimación del coste del software. La utilización del conocimiento disponible sobre estos cost drivers ha demostrado que los segmentos obtenidos tienen mayor cohesión que los conseguidos sin la utilización de dicho conocimiento, proporcionando por lo tanto unos valores de MMRE y PRED(l) superiores a sus análogos obtenidos sin el conocimiento de expertos.

- La comparación de diferentes algoritmos de clustering ha mostrado que la familia de algoritmos basados en métodos de particionado es más eficaz sobre este tipo de datos, que otros algoritmos como los de clustering jerárquico. Finalmente, el algoritmo EM, gracias a su gran versatibilidad y posibilidades de configuración ha sido el que mejores resultados ha ofrecido, siendo el utilizado en la mayoría de las pruebas realizadas.

Estas aportaciones pueden servir como punto de partida para otras aplicaciones que

utilicen el modelo propuesto.

Todos estos objetivos se han materializado en diversas publicaciones científicas. Tras la comparación de diferentes algoritmos de clustering para la segmentación de proyectos (Garre et al. 2006), se decide la utilización del algoritmo EM para el resto de pruebas.

Una vez decidido el algoritmo de clustering a utilizar, se realizan diversos trabajos consistentes en su estudio y aplicación (Garre & Charro 2005, Garre et al. 2005) sobre diferentes conjuntos de atributos de proyectos de software.

Con el fin de mejorar los resultados obtenidos se prueban nuevas aproximaciones. Una de ellas consiste en la división recursiva de los grupos de proyectos obtenidos (Garre et al. 2004). Otra posibilidad consiste en aplicar el conocimiento disponible sobre ciertos cost drivers para conducir el proceso de clustering (Cuadrado et al. 2006).

1.3. Método de trabajo

En esta sección se describe la planificación general del trabajo realizado, así como el método de evaluación seguido.

1.3.1. Planificación general

La metodología seguida en esta investigación comprende las siguientes fases:

1. **Estudio del estado de la cuestión**: Se estudian los modelos de estimación del coste del software más importantes desarrollados a lo largo de los últimos años, así como las diferentes clasificaciones en las que se han agrupado. Es necesario también estudiar las diferentes técnicas de clustering existentes, profundizando en las que son más adecuadas ante el tipo de datos a manejar.

2. **Descripción del contexto del problema**: El problema se plantea en el marco de los modelos de estimación del coste del software. Se parte de los valores, que se obtienen de MMRE y PRED(l), al aplicar un modelo de estimación paramétrico sobre la totalidad de una base de datos de proyectos. Se presenta la necesidad

de mejorar estos valores, para poder llevar a cabo un proceso de estimación más preciso y acorde con las necesidades actuales.

3. **Definición del modelo**: Se construye un nuevo modelo de estimación del coste, basándose en la combinación de modelos paramétricos y técnicas de clustering. Al aplicar clustering sobre la base de datos de proyectos, se segmenta la misma, obteniendo diferentes grupos de proyectos. Para cada uno estos grupos de proyectos, se obtiene la correspondiente función paramétrica, así como los valores de MMRE y PRED(l) que resultan de la evaluación de cada uno de los modelos a que dan lugar estas funciones.

4. **Evaluación del modelo**: Para la evaluación del modelo se realizarán varios experimentos, tomando como criterio de evaluación, los valores MMRE y PRED(l). De entre los experimentos realizados destacan:

   - Comparación de diferentes algoritmos de clustering, ya que según sea el algoritmo utilizado, los resultados obtenidos serán diferentes
   - Utilización de diferentes parámetros para la segmentación de la base de datos de proyectos
   - Análisis de la segmentación recursiva sobre proyectos
   - Estudio de la influencia de ciertos cost drivers, sobre la formación de los segmentos en el proceso de clustering

   Todos ellos realizados con el fin de mostrar las características más importantes de estos modelos paramétricos segmentados.

5. **Formulación de conclusiones**: Se concluirá con un análisis del grado de cobertura de los objetivos alcanzados, y se detallarán líneas futuras de investigación y ampliación del trabajo realizado.

La aplicación de las diferentes alternativas de segmentación de proyectos, se realizará de forma simultánea con la evaluación de las mismas, constituyendo este proceso de diseño-evaluación, el hilo conductor en el proceso de investigación.

1.3.2. Método de evaluación

El método de evaluación seguido en este trabajo consistirá en la comparación de los valores MMRE y PRED(l)[7] obtenidos, por un lado, aplicando una única función de estimación sobre la totalidad de proyectos de software, y por otro, aplicando las n-funciones que resultan de la segmentación de la base de datos de proyectos en n-segmentos.

Al ser estos unos datos totalmente objetivos, es muy sencillo realizar la comparación entre diferentes valores de ellos, mostrándose de forma clara e inmediata la bondad de una aproximación frente a la otra.

Este método de evaluación se realizará a continuación del proceso de construcción de las funciones paramétricas. Proceso que consistirá siempre en los mismos pasos, pero diferirá en el algoritmo de clustering utilizado, en los parámetros considerados dentro de cada proceso de segmentación, y en las posibles aproximaciones alternativas que se pueden utilizar para la elección de ciertos cost drivers que mejoren la segmentación realizada.

1.4. Estructura del documento

Esta memoria de tesis doctoral se estructura como se describe a continuación.

1. En el presente capítulo se recogen los objetivos y aportaciones de la tesis, y se presenta la metodología de trabajo de la investigación.

2. En el capítulo 2 se describe el estado de la cuestión en lo relativo a los diferentes modelos de estimación del coste del software existentes, y las diferentes clasificaciones que se han realizado con la intención de agruparlos según determinados criterios de afinidad en cuanto a sus características. Además se describe la técnica de clustering junto con una descripción de las familias de algoritmos existentes.

[7]Conte et al. (Conte et al. 1986) considera como aceptables, para un modelo de estimación, estas medidas de evaluación.

3. En el capítulo 3 se detalla el planteamiento del problema, analizando los actuales inconvenientes y limitaciones del uso de una única fórmula para modelizar toda una base de datos de proyectos. A la luz de las limitaciones observadas y tras exponer sus posibles soluciones, se muestra la necesidad de introducir un nuevo modelo que solvente los inconvenientes encontrados en la aproximación de una única fórmula.

4. En el capítulo 4 se describe el nuevo modelo de estimación del coste del software propuesto: modelo paramétrico segmentado.

5. En el capítulo 5 se describe la metodología general llevada a cabo para realizar la evaluación del modelo segmentado, describiendo la base de datos de proyectos utilizada, mostrando algunos algoritmos de clustering considerados, así como la selección de los parámetros de calidad predictiva seleccionados, para terminar con un conjunto de pruebas sobre la base de datos de proyectos.

6. Finalmente, en el capítulo 6 se ofrecen las conclusiones del trabajo realizado y se esbozan algunas líneas futuras de investigación relacionadas con la aplicación del modelo descrito y su posible mejora.

Capítulo 2

Estado de la cuestión

Únicamente en el movimiento, por doloroso que sea, hay vida.

Jacob Burckhardt

En este capítulo se ofrece una descripción del estado actual de las investigaciones y trabajos desarrollados en el área de la estimación del esfuerzo en el desarrollo de software. Para ello se realizará un breve recorrido por las diversas clasificaciones de los modelos que se han utilizado para realizar dichas estimaciones a lo largo de la historia, así como una descripción de los modelos más importantes. Una técnica que se ha utilizado en diferentes áreas de inteligencia artificial y estadística, pero que no se ha utilizado demasiado hasta ahora en la estimación del coste del software, es la de *clustering*. En este trabajo se investigará su aplicación en el área de Ingeniería del Software. Por lo tanto, también es necesario realizar una descripción de la clasificación de los principales algoritmos de clustering.

2.1. Introducción

Como ya se ha descrito en el primer capítulo, el desarrollo de cualquier tipo de software ha llevado asociado, desde sus inicios en los años 40, una problemática en todo lo referente a la planificación de su desarrollo en el tiempo y a la predicción de su coste.

La cancelación de proyectos, debido principalmente al incumplimiento de los plazos previstos para su realización, o la superación del presupuesto estimado, ha sido el detonante principal para que se haya investigado durante décadas sobre la forma de obtener

una información fiable, preferiblemente durante la fase de especificación de requisitos del software a desarrollar, de manera que la duración y el coste del proyecto una vez construido se ajuste lo más posible al plazo y coste previstos al inicio del mismo.

Estas investigaciones han dado lugar a multitud de modelos de estimación de software, modelos que, de forma sistemática, son capaces de ofrecer una estimación del coste del software, la cuál tendrá que ser evaluada según unos criterios determinados.

Con el fin de arrojar un poco de orden en el caos, diversos autores han clasificado estos modelos atendiendo a similitudes entre ellos.

En el presente capítulo se pretende dar una visión de las clasificaciones (sección 2.2) y de los modelos (sección 2.3) desarrollados a lo largo de la historia hasta nuestros tiempos. Además se verán los diferentes algoritmos de clustering, agrupados según su tipo (sección 2.4).

2.2. Clasificación de los Modelos de Estimación del Coste del Software

Esta sección da una visión de las clasificaciones de los modelos que se describen en la literatura sobre la estimación del coste del software. Clasificaciones que se establecen en base a las suposiciones previas y características de los modelos, permitiendo así la evaluación y comparación de los mismos.

2.2.1. Clasificación de Barry Boehm (1981)

Barry Boehm, en su libro *Software Engineering Economics* (Boehm 1981) establece una clasificación en 7 categorías, de los métodos usados para estimar el coste en el desarrollo software. Estas categorías son las siguientes:

- Modelos Algorítmicos
- Juicio de Expertos

- Analogía
- Parkinson
- Price to Win
- Top Down
- Bottom Up

Modelos Algorítmicos

Estos modelos están basados en algoritmos matemáticos, los cuales proporcionan una estimación del coste del software mediante una función de una serie de variables, conocidas como cost drivers (variables dependientes del esfuerzo que influyen significativamente sobre él).
Los modelos algorítmicos, a su vez, se pueden agrupar en:

- Modelos Lineales

  Tienen la forma

  $$Esfuerzo = a_0 + a_1 x_1 + \cdots + a_n x_n$$

  donde $x_1, \ldots, x_n$ son las variables dependientes, y $a_0, \ldots, a_n$ son los coeficientes que proporcionan un mejor ajuste para una serie de proyectos dados. Dentro de esta categoría encajarían modelos tales como SDC y Farr-Zagorski.

- Modelos Multiplicativos

  Su forma es

  $$Esfuerzo = a_0 a_1^{x_1} a_2^{x_2} \cdots a_n^{x_n}$$

  siendo $x_1, \ldots, x_n$ las variables dependientes, y $a_0, \ldots, a_n$ los coeficientes que proporcionan un mejor ajuste para los datos dados. Modelos de esta categoría serían, el modelo Walston-Felix y Doty.

- Modelos Analíticos

  Son mucho más generales que los anteriores, tomando la forma

  $$Esfuerzo = f(x_1, \ldots, x_n)$$

donde $x_1, \ldots, x_n$ son los cost drivers, y f es una función matemática no lineal ni multiplicativa. Esta categoría de modelos incluye los modelos Halstead y Putnam (Putnam 1978).

- Modelos Tabulares
  Contienen una serie de tablas que relacionan los valores de las variables dependientes, con porciones de esfuerzo de desarrollo software, o con multiplicadores usados para ajustar la estimación del esfuerzo. Son fáciles de entender y de implementar, además de ser fácilmente modificables ante la introducción de nuevas variables independientes. Ejemplos de dichos modelos son Aron, Wolverton y Boeing.

- Modelos Compuestos
  Se basan en una combinación de funciones lineales, multiplicativas, analíticas y tabulares para estimar el esfuerzo en función de las variables independientes. Tienen la ventaja de que usan la mejor forma de la función, para cada componente de la estimación del coste. Sin embargo son muy complicados de aprender y de usar a mano. Modelos de esta categoría son PRICES S, SLIM y COCOMO.

Juicio de Expertos

Las técnicas basadas en juicio de expertos llevan asociada la consulta a expertos. Dichos expertos ofrecen la experiencia que poseen en el desarrollo de proyectos de similares características a los proyectos bajo estudio, para obtener la estimación del coste de los mismos. Estas técnicas son útiles en ausencia de datos cuantificados y empíricos, pero tienen el inconveniente de que son tan buenas como lo sean las opiniones de los expertos consultados (años de experiencia no llevan asociados altos niveles de competencia). Para conseguir la mejor estimación posible, dado que pueden intervenir uno o varios expertos, es necesario utilizar técnicas de consenso tales como Delphi (Helmer 1966).

En el trabajo de Jorgensen (Jorgensen 2002), se realiza una revisión de los estudios realizados en los últimos años, en la estimación del coste del software utilizando la estimación de expertos.

Analogía

La estimación por analogía involucra la comparación del proyecto bajo estudio con una base de datos de proyectos reales, con el fin de encontrar el que más se le parezca en cuanto a sus características. De esta manera, con los datos del proyecto encontrado se puede (por analogía) realizar la estimación del coste del proyecto en desarrollo.

Parkinson

Este modelo está basado en la Ley de Parkinson, la cual establece que el coste estimado de un proyecto estará en función de todos los recursos disponibles. De esta manera, si un proyecto se debe completar en 8 meses y se dispone de 10 personas, el coste estimado para el mismo será de 80 personas-mes.

Método Price to Win

Este método dará una estimación con el objetivo de ganar el proyecto. Es decir, intentará dar el presupuesto más bajo, u ofrecer él tiempo de realización más corto. La experiencia ha demostrado que las empresas que han seguido este principio suelen cesar en sus actividades en breve.

Método de estimación Top Down

A partir de las propiedades del producto software se hace una primera estimación global, dividiéndose más tarde este coste entre los diversos componentes del sistema, sucediéndose este proceso si los componentes a su vez se dividen en otros. Normalmente este método se aplica conjuntamente con alguno de los anteriores.

Método de estimación Bottom Up

Se hace una estimación de cada uno de los componentes software básicos por separado, produciéndose a continuación la agregación de costes según se agrupan componentes sencillos en componentes más complejos. La estimación del coste del proyecto total es la suma de los costes individuales. Este método, al igual que el anterior, se suele usar en combinación con alguno de los vistos anteriormente.

2.2.2. Clasificación de DeMarco(1982)

En su libro, *Controlling Software Projects* (DeMarco 1982), DeMarco ofrece una visión de los diferentes modelos de estimación, permitiendo realizar la siguiente clasificación:

- *Corrected Single Factor Models*

  Este modelo consiste en establecer la relación entre el coste y un conjunto de parámetros en orden descendente de dependencia con respecto al mismo. Se trataría, entonces de ajustar esta curva con los datos dados. La ecuación propuesta tiene la forma:

  $$Esfuerzo = K + M_1P_1 + M_2P_2 + \cdots + M_nP_n$$

  donde P_n es el parámetro n-ésimo y M_n es su multiplicador. Normalmente la relación no suele ser lineal, por lo que la ecuación anterior toma esta otra forma:

  $$Esfuerzo = K + M_1P_1^{E_1} + M_2P_2^{E_2} + \cdots + M_nP_n^{E_n}$$

  Es necesario tener en cuenta el ruido, ignorando los parámetros cuya contribución sea substancialmente menor que el nivel de ruido. Modelos dentro de esta clasificación podrían ser los de Walston-Felix, Bailey-Basili y COCOMO.

- *Time Sensitive Cost Models*

  Este método trata de comprender la relación existente entre las personas disponibles y el tiempo de desarrollo, teniendo en cuenta que incrementar el número de personas en el desarrollo de un proyecto en un factor de X no disminuirá el tiempo en $1/X$. Modelos de este tipo serían los de Putnam (Putnam 1978) y de Parr (Parr 1980).

2.2.3. Clasificación de Conte, Dunsmore y Shen (1986)

En el libro *Software Engineering Metrics and Models* de Conte (Conte et al. 1986), los autores agrupan los modelos de estimación en cuatro categorías, dependiendo del método utilizado para derivar el modelo. Estas cuatro categorías son:

- Modelos históricos-experimentales
- Modelos estadísticos
- Modelos teóricos
- Modelos compuestos

Modelos históricos-experimentales

Esta categoría cubre la mayoría de los modelos de estimación del coste comúnmente usados hasta 1986. Consiste en la consulta a expertos sobre el esfuerzo necesario para llevar a cabo un proyecto. Estos expertos, basándose en la experiencia de proyectos similares, en la intuición y en información histórica de proyectos parecidos, ofrecen sus estimaciones. Éstas deben de ser ponderadas según el número de expertos consultados. Un ejemplo de este tipo de modelos sería el modelo TRW Wolverton.

Modelos estadísticos

Son modelos tanto lineales como no lineales, que realizan análisis de regresión para establecer la relación entre el esfuerzo y los cost drivers, tales como tamaño del proyecto. Un modelo estadístico lineal tiene la forma:

$$E = c_0 + \sum_{i=1}^{n} c_i x_i$$

donde E es el esfuerzo, c_i son constantes y x_i son los cost drivers. Modelos de esta categoría serían SDC y Farr-Zagorski.
Un modelo estadístico no lineal tiene la forma:

$$E = (a + bS^c)m(\vec{X})$$

donde S es el tamaño estimado del proyecto medido en cientos de líneas de código; a, b, y c son constantes obtenidas a partir de análisis de regresión; y $m(\vec{X})$ es un multiplicador de ajuste que depende de uno o varios de los cost drivers, denotados por el vector $\vec{X}$. Esta ecuación suele convertirse en un problema complicado de resolver, debido a que

normalmente aparecen dependencias no lineales de $m(\vec{X})$. Por lo tanto se suele utilizar un estimador nominal derivado de la ecuación anterior:

$$E_{nom} = a + bS^c$$

ajustándose este esfuerzo nominal con el factor de ajuste $m(\vec{X})$. Ejemplos de estos modelos son Walston-Felix, Bailey-Basili, y algunas de las funciones de COCOMO, y Doty.

Modelos teóricos

Se basan en la forma de razonamiento de los programadores durante el proceso de desarrollo software, así como en leyes matemáticas que sigue el proceso de desarrollo software. Ejemplos de estos modelos son el modelo Putnam, modelo Jensen, y el modelo *Halstead's Software Science*.

Modelos compuestos

Los modelos compuestos están basados en una combinación de ecuaciones analíticas, datos estadísticos (procedentes de modelos lineales y no lineales), y juicio de expertos. Modelos de este tipo serían el de Putnam, PRICE S y COCOMO.

2.2.4. Clasificación de Kitchenham 1991 y 1996

En (Kitchenham et al. 2002) Barbara Kitchenham lista cuatro métodos de estimación del coste:

- Opinión de expertos
- Analogía
- Descomposición
- Ecuaciones de estimación

El método de Descomposición implica descomponer un proyecto desde su totalidad hasta sus componentes más pequeñas (tareas de nivel inferior). La estimación total se

calcula como la suma de las estimaciones de cada una de las tareas de nivel inferior. Los otros tres métodos son iguales a los descritos por Boehm (Boehm 1984) (las Ecuaciones de estimación se corresponden con los Modelos algorítmicos).

En su posterior clasificación, Kitchenham ya no menciona el método de Descomposición, sin embargo se listan dos nuevas formas de Ecuaciones de estimación (Modelos de Coste), Ecuaciones Matemáticas y Tablas Look-Up.

2.2.5. Clasificación de Fairley 1992

En el trabajo de Fairley (Fairley 1992), el autor clasifica las técnicas de estimación del coste en las tres categorías siguientes:

- Modelos empíricos
- Modelos basados en Regresión
- Modelos basados en la Teoría

Modelos empíricos

Se derivan de las observaciones de rendimiento pasadas, e incluyen reglas del pulgar, analogía y puntos de función.

Modelos basados en Regresión

Como por ejemplo COCOMO, proceden de datos históricos y expresan la relación matemática entre las variables del proyecto. Se construyen mediante ecuaciones de regresión, que se obtienen a partir de los datos de proyectos desarrollados. A veces es necesario aplicar factores de ajuste, a los valores calculados por las ecuaciones, para obtener unas estimaciones finales del esfuerzo y del tiempo.

Modelos basados en la Teoría

Proceden de consideraciones teóricas subyacentes que caracterizan algunos aspectos del proceso de desarrollo software. Ejemplos de estos modelos son SLIM, y PRICE S.

2.2.6. Clasificación de Walkerden y Jeffery 1997

En Walkerden (Walkerden & Jeffery 1997) los modelos se clasifican de acuerdo al método de predicción en el que se basan. Los modelos empíricos se clasifican en paramétricos o no-paramétricos. Un modelo paramétrico tiene forma de función explícita, relacionando una variable dependiente con una o varias variables independientes (p.e. COCOMO). Un modelo no-paramétrico no tiene una forma de función explícita (p.e. un modelo desarrollado a partir de una técnica de aprendizaje automático, tal como una red neuronal artificial). La clasificación sería la siguiente:

- Modelos empíricos paramétricos
- Modelos empíricos no-paramétricos
- Modelos analógicos
- Modelos teóricos
- Modelos heurísticos

Los autores establecen que cualquier predicción que relacione el atributo de interés con cualquier otro atributo, debe de basarse en un modelo empírico. Este modelo empírico sería el punto de partida para cada uno de los métodos de predicción. Juicio de expertos es un método de predicción según Boehm (Boehm 1981). Sin embargo, no se incluye en esta clasificación, ya que según los autores, juicio de expertos no es fácilmente caracterizable y solamente se utiliza cuando hay expertos disponibles.

Modelos empíricos paramétricos

Es la clasificación más común de modelos. En su forma más simple, un modelo empírico paramétrico sería una función que relaciona el esfuerzo de desarrollo de un sistema, con una medida de tamaño. Modelos de esta categoría serían FPA, Basic COCOMO, Walston-Felix y Bailey-Basili. Estos modelos son *modelos de esfuerzo*. Existe otra forma de estos modelos, llamada *elapsed time models*, que se centra en la relación existente entre el esfuerzo y la duración del proyecto. Ejemplos de modelos de esta categoría son Putnam y Parr.

Modelos empíricos no-paramétricos

Como se ha comentado anteriormente, estos modelos no tienen una forma funcional, y proceden de diversas técnicas de aprendizaje automático (*machine learning*). Un ejemplo podría ser la técnica OSR *Optimized Set Reduction* (Briand et al. 1992), consistente en una aproximación de reconocimiento de patrones para analizar los datos de un árbol de decisión. Otro método no-paramétrico se describe en el trabajo de Srinivasan (Srinivasan & Fisher 1995), el cual usa back-propagation para entrenar una red neuronal.

Modelos analógicos

La estimación se realiza mediante interpolación, a partir de los datos de proyectos análogos previamente desarrollados. Ejemplos de estos modelos son ESTOR (Mukhopadhyay et al. 1992) y la herramienta de estimación ANGEL (Shepperd & Schofield 1997).

Modelos teóricos

Proponen un modelo numérico basándose en un modelo empírico. El modelo teórico debe de validarse empíricamente. Uno de estos modelos ha sido desarrollado por Abdel (Abdel-Hamid & Madnick 1986). Este modelo se basa en relaciones dinámicas de retroalimentación entre gestión de personal, producción de software, planificación y control.

Modelos heurísticos

Los modelos heurísticos se usan como una extensión a los anteriores, pudiéndose utilizar para ajustar las estimaciones realizadas por otros métodos. No existen, sin embargo, modelos que se basen en los modelos heurísticos como base principal.

2.2.7. Clasificación realizada por Barry Boehm, Chris Abts y Sunita Chulani en el año 2000

Tras la primera clasificación de Boehm (Boehm 1981), surgieron nuevos modelos que se recogen en este nuevo trabajo. En el trabajo de Boehm (Boehm, Abts & Chulani 2000) los autores presentan la siguiente clasificación de los modelos existentes hasta ese año:

- Técnicas basadas en Modelos

  Se corresponden con los Modelos Algorítmicos vistos en 2.2.1. Dentro de estos modelos, Boehm incluye a SLIM, Checkpoint, PRICE-S, ESTIMACS, SEER-SEM, SELECT Estimator, y COCOMO II.

- Técnicas basadas en Expertos

  Igualmente, se corresponden con los modelos basados en el Juicio de Expertos vistos también en 2.2.1. Dentro de éstos se tendría la Técnica Delphi y Work Breakdown Structure.

- Orientados al Aprendizaje

- Modelos Dinámicos

- Modelos basados en Técnicas de Regresión

- Modelos Compuestos-Bayesianos

Orientados al Aprendizaje

Tienen su origen en los modelos por Analogía (ver 2.2.1), referida por los autores (Boehm, Abts & Chulani 2000) como Estudio de Casos (*Case Studies*), y Razonamiento Basado en Casos (*Case Based Reasoning CBR*) por Shepperd (Shepperd & Schofield 1997). Se incluyen también en esta clasificación las estimaciones realizadas utilizando Redes Neuronales.

Estudio de Casos Representa un estudio inductivo, por el cual los estimadores intentan aprender mediante la extrapolación de ejemplos específicos dados. Se estudian las condiciones del entorno, restricciones, decisiones técnicas y de gestión tomadas, así como el final acontecido de una serie de proyectos previamente realizados, con el fin de extraer los vínculos subyacentes entre causa y efecto, de manera que se puedan aplicar en el nuevo contexto. Para realizar estimaciones de nuevos proyectos, se buscan proyectos de similares características, con la creencia de que (por analogía) proyectos similares es muy probable que se comporten de la misma forma (en cuanto a costos y planificación

en el tiempo). Los proyectos de los que se extrapola la información, pueden haber sido realizados por la propia organización, o procedentes de entidades externas. La primera forma sería la más deseable, aunque si el proyecto está bien documentado y procede de una organización que realiza actividades similares a la nuestra, también puede ser beneficioso.
Para mayor información ver sección 2.2.9.

Redes Neuronales Se describen en detalle en la sección 2.2.9.

Modelos Dinámicos

Estos modelos reconocen explícitamente que el esfuerzo de proyectos de software o los factores de coste, cambian durante el desarrollo del sistema, esto es, son dinámicos en el tiempo. Esto introduce una característica novedosa, ya que hasta el momento todos los modelos eran estáticos, realizando la estimación basándose en una instantánea de la situación en un instante de tiempo dado. La técnica dinámica más destacada se basa en la aproximación dinámica del sistema (*System Dynamics Approach*), desarrollada por Jay Forrester hace unos cuarenta y cinco años (Forrester 1961).

System Dynamics Approach Se basan en modelos de simulación continua, representando el modelo resultante como un grafo de información que va cambiando a lo largo del tiempo. Matemáticamente, los modelos de simulación se representan mediante un conjunto de ecuaciones diferenciales de primer orden (Madachy 1994):

$$x'(t) = f(x, p)$$

donde x es un vector que describe los niveles o estados del modelo, p es el conjunto de parámetros del modelo, f una función vectorial no lineal, y t representa el tiempo.

Esta técnica se ha aplicado satisfactoriamente durante los últimos quince años. Abdel-Hamid ha construido varios modelos que predicen cambios en el coste del proyecto, las necesidades de personal y la planificación en el tiempo (Abdel-Hamid 1989*a*, Abdel-Hamid 1989*b*, Abdel-Hamid 1993, Abdel-Hamid & Madnick 1991). Así mismo, este autor

también ha realizado estudios acerca de la reusabilidad del software (Abdel-Hamid & Madnick 1993) utilizando esta aproximación.

Modelos basados en Técnicas de Regresión

Estas técnicas son las más utilizadas en la construcción de modelos. Se suelen utilizar junto con las técnicas basadas en modelos, e incluyen *Regresión Standard* y *Regresión Robusta*.

Regresión Standard Utiliza la aproximación estadística clásica de mínimos cuadrados. Se basa en el método de Mínimos Cuadrados Ordinario (OLS *Ordinary Least Squares*), discutido en libros como el de Judge y Weisberg (Judge et al. 1993, Weisberg 1985). La razón de su gran utilización reside en la sencillez de manejo y su simplicidad. Está disponible en multitud de paquetes estadísticos comerciales, tales como Minitab, SPlus, SPSS, etc.
La forma típica de uno de estos modelos es:

$$y_t = \beta_1 + \beta_2 x_{t2} + \cdots + \beta_k x_{tk} + e_t$$

donde $x_{t2} \cdots x_{tk}$ son los indicadores para la observación t_{th}, $\beta_2 \cdots \beta_k$ son los coeficientes de respuesta, β_1 es un parámetro de corte e y_t es la variable respuesta para la observación t_{th}. El término de error e_t es una variable aleatoria con distribución de probabilidad normal en la mayoría de los casos. El método OLS funciona estimando los coeficientes de respuesta y el parámetro de corte, minimizando el término de error de mínimos cuadrados r_i^2 donde r_i es la diferencia entre la respuesta observada y la respuesta ofrecida por el modelo para la observación i_{th}. De esta manera todas las observaciones tienen una influencia equivalente en la ecuación del modelo, excepto los *outliers*[1] (puntos muy diferentes al resto) que influirían de forma despreciable sobre el mismo.

El método OLS es adecuado cuando hay muchos datos disponibles, no hay atributos nulos, no hay outliers, y los indicadores no están correlacionados, tienen una interpretación sencilla cuando se usan en el modelo y son continuos.

[1]Se usará el término inglés.

Una variación del modelo anterior se usó para calibrar la versión 1997 de COCOMO II. Se utilizó regresión múltiple para estimar los coeficientes b asociados con los 5 factores de escala y los 17 multiplicadores de esfuerzo. Uno de los resultados obtenidos fue por ejemplo, que desarrollar software con vistas a ser reutilizado es más barato que desarrollarlo a medida para una única aplicación.

Regresión Robusta Es una mejora sobre la aproximación OLS standard. La regresión robusta mitiga el problema de los outliers. Dentro de los datos de los proyectos de software existen multitud de outliers, debidos a la falta de uniformidad que existe a la hora de definir las métricas de software, junto con la coexistencia de varios procesos de desarrollo software y la disponibilidad de datos cualitativos más que cuantitativos. Existen numerosas técnicas estadísticas que encajan dentro de esta categoría de Regresión Robusta.

La mayoría de los modelos paramétricos de coste (COCOMO II, SLIM, Checkpoint, etc.) usan alguna forma de técnica de regresión, debido a su simplicidad y amplia aceptación.

Modelos Compuestos-Bayesianos

Todos los métodos examinados anteriormente tienen pros y contras, se trata de aprovechar las ventajas de cada uno de ellos mediante combinaciones de los mismos, de manera que se obtenga la mejor estimación posible.

Aproximación Bayesiana Una forma curiosa de estimar que ha sido utilizada para el desarrollo del modelo COCOMO II es el análisis Bayesiano tal y como describe Chulani (Chulani 1998, Chulani et al. 1998).

El análisis Bayesiano es una forma de razonamiento inductivo que ha sido utilizada en diversas disciplinas científicas. Una de las características de la aproximación Bayesiana es que permite utilizar de una forma consistente, datos obtenidos a partir de bases

de datos de proyectos históricos, así como la experiencia de expertos (conocimiento a priori) para realizar inferencias. Esto se consigue a través del teorema de Bayes, el cual proporciona unos datos a posteriori que representan los parámetros del modelo. Este proceso de transformación de datos a priori en datos a posteriori, se puede ver como un proceso de aprendizaje. Usando el teorema de Bayes, se pueden combinar los dos tipos de información de la siguiente forma:

$$f(\beta/Y) = \frac{f(Y/\beta)f(\beta)}{f(Y)}$$

donde β es el vector de parámetros de interés, e Y es el vector de observaciones procedente de la función de densidad a posteriori $f(\beta/Y)$. La información proporcionada viene dada por $f(Y/\beta)$, y $f(\beta)$ es la información a priori que engloba toda la información de los expertos acerca de β. La anterior ecuación se puede escribir como:

$$f(\beta/Y) \propto l(\beta/Y)f(\beta)$$

lo que significa

$$Posteriori \propto Muestra * Priori$$

En el contexto Bayesiano, las probabilidades a priori son las probabilidades incondicionales asociadas con la información de muestra, mientras que las probabilidades a posteriori son las probabilidades condicionales dadas por la información a priori.

La aproximación Bayesiana ha sido utilizada en la calibración del modelo COCOMO II realizada en el año 2000. Calibración que mejora la realizada en el año 1997 con modelos de regresión.

El análisis Bayesiano tiene todas las ventajas de la regresión standard, y además incluye el conocimiento a priori que proporcionan las opiniones de diversos expertos. Los datos disponibles en ingeniería del software, normalmente son escasos e incompletos, siendo complicado el poder hacer buenas estimaciones. Esto queda remediado, en gran medida, al aportar información procedente de expertos, obteniéndose así un modelo de estimación más robusto.

2.2.8. Clasificación de Wieczorek 2001

En la obra de Wieczorek (Wieczorek & Briand 2001), los autores realizan una clasificación de los modelos de estimación del coste en base a una serie de definiciones de los términos *modelo de estimación*, *método de estimación* y *técnica de estimación*. A continuación se muestran dos definiciones de *método de estimación*:

I. Un *método de estimación* consiste en una o más técnicas de estimación. Una *técnica de estimación* es un procedimiento para obtener una estimación del coste directamente del conocimiento de expertos y/o datos de proyecto. Una *estimación* se obtiene aplicando directamente una o más *técnicas de estimación* sobre los datos disponibles.

II. Un *método de estimación* está formado por uno o varios *modelos de estimación*, posiblemente un *método de modelado* y un *método de aplicación* para aplicar los *modelos de estimación*. Un *modelo de estimación* se construye a partir de una o más *técnicas de modelado* de acuerdo a los *métodos de modelado*. Una *técnica de modelado* es un procedimiento sencillo para construir un *modelo de estimación* a partir de los datos disponibles y/o el conocimiento de expertos. Una *estimación* se obtiene entonces, aplicando uno o más *modelos de estimación* en un contexto específico, de acuerdo a un *método de modelo de aplicación*.

La clasificación ofrecida por Wieczorek y Briand (Wieczorek & Briand 2001) se puede ver en la figura 2.1.

Métodos Basados en Modelos

Un método de estimación basado en modelos, en general, lleva asociado un método de modelado, un modelo, y un método de modelo de aplicación. Un modelo de estimación del esfuerzo toma una serie de entradas (factores de productividad y una estimación del tamaño del sistema) y produce un valor concreto o una distribución del esfuerzo.

Modelos Genéricos Estos métodos de estimación generan modelos que se pueden aplicar a proyectos de diferentes contextos.

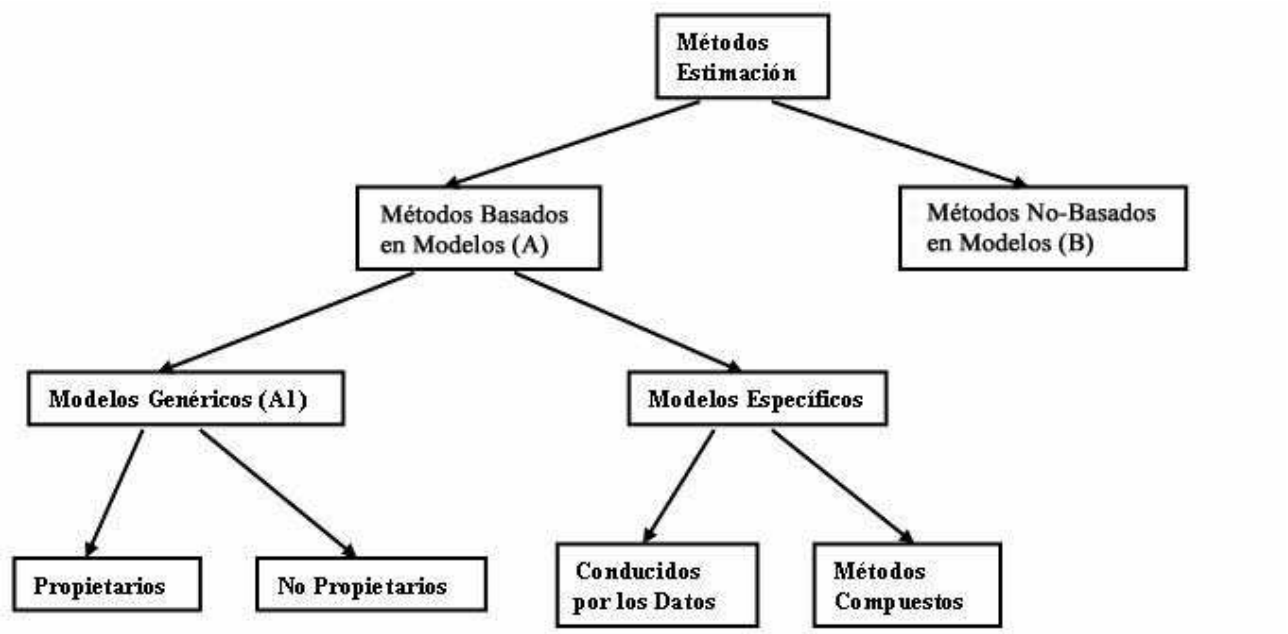

Figura 2.1: Clasificación de Modelos de Estimación según (Wieczorek & Briand 2001)

- Propietarios

  Los modelos y métodos de modelado no son públicos, y por lo tanto no están completamente documentados.

- No Propietarios

  Los modelos y métodos de modelado son de dominio público, y por lo tanto completamente documentados. Ejemplos de estos métodos serían: COCOMO, COCOMO II, SLIM.

Modelos Específicos Estos modelos incluyen a los modelos locales, únicamente válidos en el contexto para el cual han sido desarrollados.

- Conducidos por los Datos

  Están basados en el análisis de datos, obteniéndose los modelos a partir de los datos. Se puede a su vez, distinguir entre modelos paramétricos y no-paramétricos. Los modelos paramétricos necesitan la especificación a priori de una relación

funcional entre diversos atributos del proyecto y el coste. A partir de ahí, el modelo intenta encajar los datos disponibles sobre la función dada mediante regresión. Los modelos no-paramétricos obtienen modelos que no hacen ninguna suposición a priori sobre la forma de la función que relaciona atributos de proyecto y coste (aunque de alguna forma siempre es necesario realizar alguna suposición). Algunos métodos de este tipo serían: CART (*Classification and Regression Trees*), OSR (*Optimized Set Reduction*), Stepwise ANOVA (*Stepwise Analysis of Variance*), OLS (*Ordinary Least Squares Regression*) y Regresión Robusta.

- Métodos Compuestos

  Los modelos de estimación se construyen a partir de la combinación de la opinión de expertos y la utilización de técnicas conducidas por los datos. Ejemplos de métodos compuestos serían: COBRA (*Cost Estimation Benchmarking and Risk Analysis*) y Métodos basados en analogía.

Métodos No-Basados en Modelos

Consisten en una o varias técnicas de estimación, junto con una especificación de como aplicarlas en un cierto contexto. Estos métodos no involucran la construcción de ningún modelo, ya que realizan directamente la estimación.

Estos modelos suelen utilizar la opinión de expertos para obtener una primera estimación subjetiva del esfuerzo, obteniendo el esfuerzo final tanto por aproximación top-down como bottom-up.

2.2.9. Estado actual

Se ha observado como desde el año 1981 hasta el 2001 se han sucedido diversas clasificaciones de los modelos de estimación del coste del software, según diferentes autores. Desde el último trabajo comentado en este documento, de Wieczorek (Wieczorek & Briand 2001), no se han realizado nuevas clasificaciones, sin embargo sí que han aparecido numerosos trabajos aplicando nuevas técnicas y combinaciones de ellas, para mejorar el proceso de estimación del coste del software. La mayoría de estas técnicas son métodos no algorítmicos, que se habían utilizado en diferentes campos para propósitos diversos,

pero que nunca se habían utilizado en el de Ingeniería del Software. Dichas técnicas son las enmarcadas dentro del Aprendizaje Automático (*Machine Learning*), las cuales tratan de *aprender*, a partir de bases de datos de proyectos, la relación que existe entre los diferentes atributos de un proyecto de software, con el coste del mismo, para poder más tarde establecer dicha relación ante nuevos proyectos. Algunas de estas técnicas ya han sido incluidas en clasificaciones anteriores, como la de Boehm (Boehm, Abts & Chulani 2000) en la sección 2.2.7, otras no habían aparecido anteriormente:

- Razonamiento Basado en Casos - CBR
- Redes Neuronales Artificiales
- Sistemas Fuzzy
- Algoritmos Genéticos - Programación Genética
- Inducción de Reglas
- CART - Classification and Regression Trees

Desde la clasificación de Boehm (Boehm, Abts & Chulani 2000) han surgido nuevos trabajos, como hemos indicado anteriormente, por ello a continuación se hará una breve descripción de los más destacados, junto con una breve explicación de las técnicas utilizadas.

Razonamiento Basado en Casos - CBR

La idea de utilizar analogías para estimar el coste de desarrollo software no es nueva. Boehm (Boehm 1981) sugiere esta idea, y más tarde fue reiterada por Cowderoy y Jenkins (Cowderoy & Jenkins 1988), pero de nuevo sin ningún mecanismo formal para poder seleccionar las analogías. El siguiente desarrollo fue llevado a cabo por Vicinanza (Mukhopadhyay et al. 1992, Vicinanza et al. 1990), el cual sugirió que los avances llevados a cabo en CBR dentro del campo del Aprendizaje Automático deberían adaptarse para poder realizar buenas predicciones de coste del software. Los trabajos más destacados fueron llevados a cabo por Shepperd (Shepperd et al. 1996, Shepperd & Schofield 1997).

La idea, *proyectos de software similares tienen un coste similar*, se desarrolla como sigue a continuación. Dados n proyectos finalizados, se considera cada uno de ellos como un caso separado que se añade a una base de casos. Cada caso se caracteriza por p características que pueden ser continuas (experiencia del director de proyecto), discretas (número de interfaces), o categóricas (entorno de desarrollo). Además es necesario conocer la característica que se desea predecir. El proyecto para el cual se desea realizar la estimación (conocido como caso objetivo) debe de caracterizarse de la misma forma que los proyectos en la base de datos (en términos de las p características). A continuación se realizaría la medida de similitud entre el caso objetivo y los casos de la base de casos, para ello se dibujaría en el espacio p-dimensional el caso objetivo y se procedería, mediante una forma modificada de la distancia euclídea, a identificar el caso más parecido. De esta manera se obtendrían los valores a utilizar para realizar la predicción. Finalmente se añadiría a la base de casos el caso objetivo solucionado.

De forma resumida, CBR consiste en los siguientes cuatro pasos:

1. Caracterización de casos
2. Almacenamiento de casos pasados
3. Recuperación de casos similares para ser usados como analogías, utilizando una variante de la distancia euclídea
4. Utilizar el caso recuperado para resolver el caso objeto de estudio.

Para mayor información sobre el proceso consultar a Leake (Leake 1996).

Existen dos herramientas para estimar mediante analogía, ESTER (Mukhopadhyay et al. 1992) y ANGEL (Shepperd & Schofield 1997).

ANGEL es una herramienta de estimación basada en analogía que busca las similitudes entre una entidad objetivo, tal como un proyecto de software, y un conjunto de entidades históricas de la misma clase. Cada entidad se caracteriza por una serie de

atributos que están disponibles en el momento en que se necesita realizar la estimación. Las similitudes se miden mediante distancia euclídea en el espacio p-dimensional.

Los sistemas de estimación del coste basados en analogías tienen una importante limitación, debida al uso de datos categóricos tales como *muy bajo, bajo, alto, muy alto*. Estas calificaciones de los atributos se conocen como valores lingüísticos en terminología de lógica difusa. Una aproximación para tratar con estos términos reside en la utilización de la teoría de conjuntos difusos, tal y como explica Zadeh (Zadeh 1965). Por lo tanto, la unión de CBR y Sistemas de Lógica Difusa combina las ventajas de ambos para afrontar los problemas de estimación del coste del software. Trabajos que han afrontado problemas de estimación del coste de esta forma son, por ejemplo, los de Ali Idri (Idri & Abran 2002, Idri et al. 2001).

A continuación se presentan algunos trabajos realizados por diferentes autores sobre CBR. Muchos experimentos se han centrado en comparar la precisión en la estimación de la aproximación CBR frente a otras técnicas de estimación. Los resultados obtenidos han mostrado que las estimaciones basadas en analogía no superan a otras técnicas en todas las situaciones. Por ejemplo Shepperd, Niessink y Kadoda (Shepperd et al. 1996, Niessink & Van Vliet 1997, Dadoda et al. 2000, Shepperd & Schofield 1997) han demostrado que las estimaciones basadas en analogía ofrecen mejores resultados que los modelos de regresión lineal. Por otra parte Briand y Myrtveit (Briand et al. 2000, Myrtveit & Stensrud 1999) han llegado a conclusiones contrarias a éstas.

Mukhopadhyay (Mukhopadhyay et al. 1992) compararon el rendimiento de su herramienta Estor con juicio de expertos, puntos de función y COCOMO, usando el conjunto de datos de Kemerer. Las conclusiones fueron que los expertos superaban a Estor, que por contra, superaba a los otros dos.

Finnie (Finnie et al. 1997) compararon los modelos por analogía con puntos de función basados en modelos de regresión y redes neuronales. Los datos usados fueron 299 proyectos de 17 organizaciones diferentes. El rendimiento del modelo por analogía fue superior a los modelos de regresión, siendo inferior al obtenido con redes neuronales.

Briand (Briand et al. 1999) compararon el método de regresión stepwise ANOVA, el de árboles de regresión CART, y ANGEL. Además se hicieron estudios combinando CART con ANOVA y CART con ANGEL. En general, los métodos basados en analogía fueron significativamente peores que los otros métodos usados, y la combinación de los métodos no mejoró los resultados.

Stephen G. MacDonell y Martin J. Shepperd comparan (MacDonell & Shepperd 2003) el uso combinado de CBR, juicio de expertos y regresión lineal aplicados a un sistema médico, observando que cuando una técnica ofrece unos resultados pobres, las otras los mejoran significativamente.

Redes Neuronales Artificiales

De acuerdo a Gray y McDonell (Gray & McDonell 1997), las redes neuronales son la forma de estimación de software basada en *construcción del modelo*, más comunmente utilizada como alternativa a la regresión de mínimos cuadrados. Consisten en modelos de estimación que se "entrenan" usando datos históricos, para producir cada vez mejores resultados mediante el ajuste de los valores de sus parámetros, minimizando el error cometido (diferencia entre los valores esperados y los de salida). Según estos autores (Gray & McDonell 1997), la forma más común de red neuronal en el campo de ingeniería del software es la red neuronal entrenada con propagación hacia atrás (*backpropagation trained feed-fordward*, figura 2.2).

Una red neuronal está formada por nodos (neuronas) organizados por capas (de entrada, intermedias y de salida) y unidos mediante enlaces entre ellos. Las neuronas de la capa de entrada aceptan los parámetros del proyecto, mientras que la neurona de la capa de salida ofrece la estimación del coste esperado. Los enlaces entre neuronas tienen asociados pesos y cada neurona en la red calcula una función no-lineal de su entrada. Estas funciones se conocen como funciones de activación. La forma más común de función de activación es:

$$\frac{1}{1 + e^{(-\sum W_i I_i)}}$$

donde $\sum W_i I_i$ es una suma ponderada de las entradas, I_i, a la neurona i.

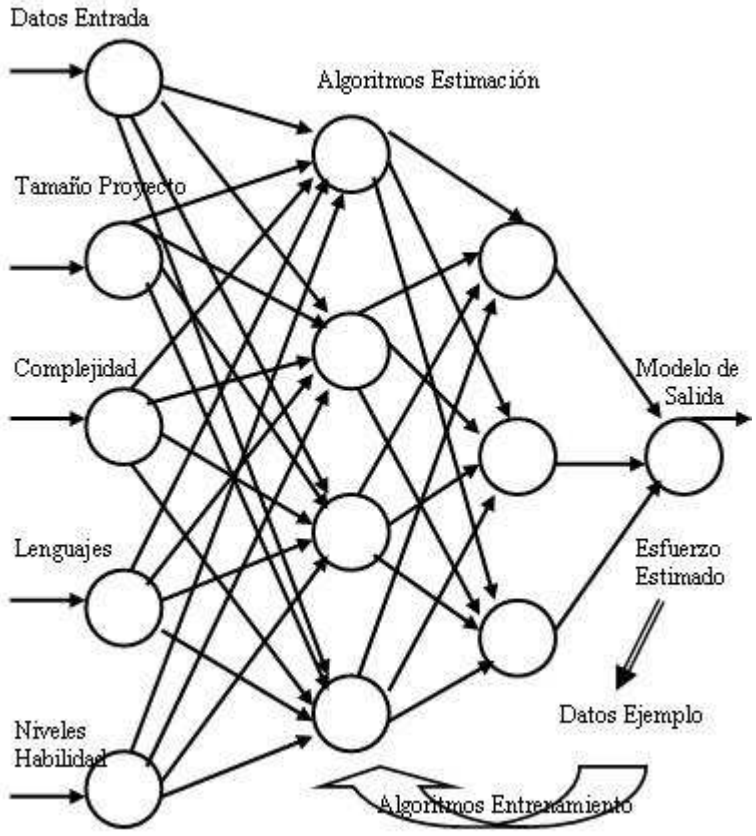

Figura 2.2: Modelo de Estimación mediante una Red Neuronal

Para construir el modelo neuronal es necesario establecer el número de capas, así como el número de nodos por capa, y la forma en la que los nodos se enlazan. También hay que establecer los valores iniciales de los pesos de los enlaces entre nodos, y el algoritmo específico de entrenamiento que se usará, normalmente backpropagation, dependiendo de estos factores, los resultados obtenidos pueden variar enormemente. Una vez que el modelo de red está establecido, es necesario entrenarla con una serie de datos de ejemplo, en este caso una serie de proyectos previamente realizados. Las neuronas de la capa de entrada se alimentan con el conjunto de variables de entrada, por ejemplo puntos de función, número de pantallas, número de ficheros, etc. La red itera un gran número de veces, ajustando los valores de los pesos de los enlaces, en función de los errores cometidos, hasta que la diferencia entre los valores de salida y los esperados está dentro

de un umbral de error establecido (valor delta). La especificación de este valor delta, también es algo importante, ya que sin él, el modelo se podría entrenar hasta "aprender de memoria" (*overtrained*) los ejemplos suministrados, realizando estimaciones perfectas para casos pertenecientes a la base de datos de proyectos de ejemplo, pero realizando pobres estimaciones para proyectos desconocidos.

Wittig (Wittig 1995) obtuvo precisiones del 10 % para un modelo de este tipo al ser usado para estimar el esfuerzo de desarrollo software. Pero cuidado, ya que tiene los mismos problemas estadísticos con los datos de entrenamiento, que las técnicas de regresión. En particular, se necesitan grandes conjuntos de datos para entrenar de forma apropiada las redes neuronales.

Una desventaja importante de las redes neuronales es que son como cajas negras, no permitiendo realizar análisis de los datos obtenidos, es decir, no se conoce el proceso seguido para conseguir esos datos, y por tanto, no contribuyen a incrementar el conocimiento humano del área al que son aplicadas.

En la página web de NeuroSolutions[2] se puede encontrar información completa sobre Redes Neuronales.

Durante los últimos años se han realizado numerosos estudios, utilizando redes neuronales, para la estimación del coste del software. Sobre todo, estudios en los que se compara la precisión obtenida utilizando redes neuronales con la obtenida al utilizar otros métodos de estimación. A continuación se verán algunos de estos trabajos.

Shukla (Shukla 2000) usa algoritmos genéticos para entrenar una red neuronal, en vez de realizar el entrenamiento tradicional. Con esta nueva forma de aprendizaje de la red, se obtienen predicciones de la estimación del esfuerzo más precisas que si se usa backpropagation para su entrenamiento. Así mismo los resultados obtenidos también mejoran los obtenidos al utilizar un árbol de regresión para realizar la estimación.

Heiat (Heiat 2002), compara el uso de una red neuronal con el análisis de regresión. En

[2]http://www.nd.com/neurosolutions/products/ns/whatisNN.html.

este estudio se concluye que cuando se usan de forma combinada, lenguajes de tercera y cuarta generación, la red neuronal produce mejores resultados que el análisis de regresión convencional, en términos de error medio.

Por último, mencionar la utilización de sistemas difusos con redes neuronales en el trabajo de Gray y McDonell (Gray & McDonell 1997).

Sistemas Fuzzy

Los sistemas de lógica difusa (*Fuzzy Logic FL*), desde su establecimiento por Zadeh en 1965 (Zadeh 1965), han sido objeto de numerosos estudios, utilizándose a partir de los años 90 en varios campos, como por ejemplo robótica, medicina y procesamiento de imágenes. En estimación del esfuerzo software, no se han utilizado hasta hace relativamente poco tiempo. Por ejemplo se pueden ver los trabajos realizados por Ali Idri en analogía difusa (Idri & Abran 2002, Idri et al. 2001), o los de Gray (Gray & McDonell 1997), en los que se combinan los puntos fuertes de las redes neuronales y los sistemas difusos. Más recientemente, M.A. Sicilia et. al (Sicilia et al. 2005), aplican la teoría difusa sobre las entradas a los modelos paramétricos de estimación del coste, además Crespo et. al (Crespo et al. 2003, Crespo et al. 2004) también abordan este tema desde otros puntos de vista.

A continuación, se comentan brevemente las bases teóricas de las partes de la lógica difusa que se usan en la estimación del coste del software.

Un sistema difuso está formado por tres componentes principales, funciones de pertenencia (asociadas a conjuntos difusos), una base de reglas, y un proceso de concreción (*defuzzification*). Un conjunto difuso es un conjunto con una función de pertenencia μ, que toma valores en el intervalo real [0, 1]. Esta definición extiende la de la teoría de conjuntos clásica (conjuntos *crisp*) donde la función de pertenencia toma los valores 0 o 1. Los conjuntos difusos se pueden usar de forma muy eficiente para representar valores lingüísticos tales como *bajo, joven, o complejo* (Jager 1995). En la teoría de conjuntos clásica, un individuo se considera joven si tiene una edad comprendida entre 21 y 32 años, por ejemplo, no siéndolo en el resto de casos. Pero en la realidad, qué sucede con

una persona que tiene 33, 34, o 20 años, ¿ya no es joven? La teoría de conjuntos difusos ofrece una solución bastante cercana a la forma del razonamiento humano, una persona puede ser joven entre 21 y 32 años, pero no deja de serlo por tener 33 años, sino que el grado en el que pertenece al conjunto de personas jóvenes será algo menor que 1, por ejemplo 0,9.

Por lo tanto, el valor lingüístico joven (para el atributo edad), se podría representar de tres formas diferentes: mediante un conjunto difuso 2.3 a), mediante conjunto crisp 2.3 b), o por un valor numérico 2.3 c). La representación mediante un conjunto difuso es mucho más adecuada que las otras dos, ya que:

- Es más general
- Es la forma en la que las personas interpretan el valor lingüístico
- La transición de un valor lingüístico a un valor lingüístico continuo es gradual

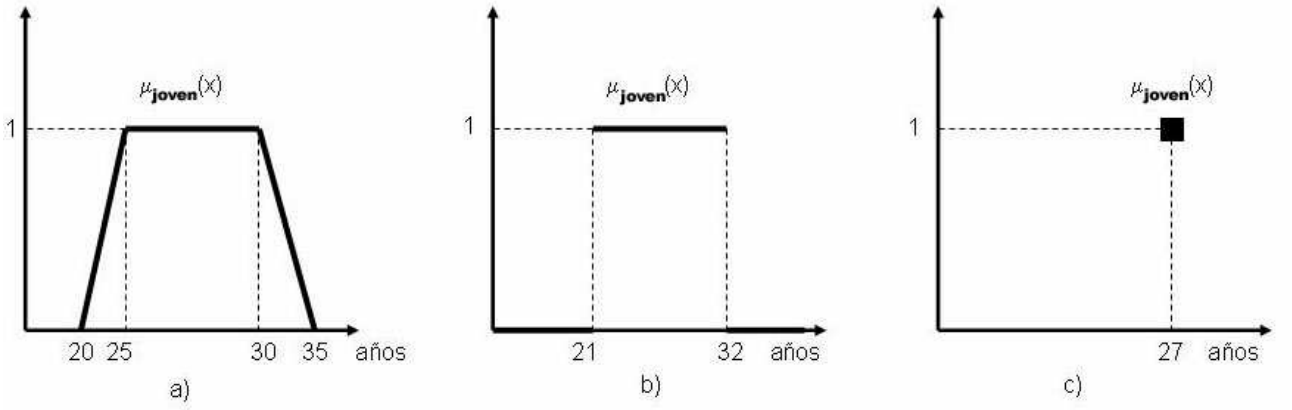

Figura 2.3: a)Representación difusa del valor lingüístico joven. b) Representación clásica. c) Representación numérica

La segunda componente, la base de reglas, se extrae a partir del conocimiento de expertos sobre el dominio del problema en cuestión. Estas reglas son las que establecen los valores de las funciones de pertenencia para un valor de x dado, por ejemplo una edad de 33 tiene un grado de pertenencia de 0,9 y no de 0,85, para el conjunto de personas jóvenes.

El tercer componente, *defuzzification process*, asocia las salidas obtenidas con una etiqueta o valor numérico. Por ejemplo, una edad de 18 años llevaría asociada la etiqueta de *muy joven*. Para más información sobre sistemas difusos se puede consultar el trabajo de Munakata (Munakata & Jani 1994).

Algoritmos Genéticos - Programación Genética

Durante los últimos treinta años se ha experimentado un creciente interés por la construcción de sistemas de resolución de problemas basados en el proceso de evolución natural. En este sentido los algoritmos genéticos son algoritmos de búsqueda y optimización cuyo mecanismo de funcionamiento está basado en dicho proceso. Es bien conocido por los biólogos que el proceso de evolución natural tiene lugar en los cromosomas, que poseen la información genética de los individuos.

John Holland, en los años 70, fue el primero que trasladó esta estrategia de actuación al campo de la computación con el fin de diseñar sistemas artificiales con un comportamiento análogo al del sistema evolutivo natural. Este intento trajo consigo el nacimiento de lo que hoy se conoce como algoritmos evolutivos. Un algoritmo evolutivo es un algoritmo probabilístico que mantiene una población de individuos $P(t)$ para cada iteración t. Cada uno de los individuos constituye una solución potencial del problema, representada mediante alguna estructura de datos. Cada una de estas potenciales soluciones es evaluada para obtener una medida de su calidad como solución. Posteriormente, se construye una nueva población (iteración $t + 1$) mediante un proceso de selección en $P(t)$, de manera que los mejores individuos tienen mayor probabilidad de ser seleccionados. A continuación se altera esta nueva población mediante la aplicación sobre algunos miembros de la misma de ciertos operadores genéticos. Estos operadores genéticos pueden ser unarios (mutaciones), produciendo nuevos individuos mediante pequeños cambios en el individuo sobre el que se aplican, o de aridad superior (cruces), de manera que dos o más individuos se combinan para producir nuevos individuos (reproducción). Así, se obtiene finalmente una nueva población $P(t + 1)$. Tras un número de iteraciones o generaciones, es de esperar que el mejor individuo de la población final

obtenida represente una solución del problema óptima o al menos razonablemente buena.

Un caso concreto de algoritmos evolutivos son los denominados algoritmos genéticos. Estos algoritmos se caracterizan por utilizar operadores de cruce en el proceso de alteración de la población. Teniendo en cuenta lo anterior, los conceptos básicos que se manejan en la teoría de los algoritmos genéticos son los siguientes:

- **Individuo:** Un individuo es una potencial solución del problema que se pretende resolver mediante el algoritmo genético.
- **Población:** Conjunto de individuos con los que se trabaja en el algoritmo genético en una determinada iteración. En un algoritmo genético los individuos que constituyen la población cambian de una generación a la siguiente, pero generalmente, el tamaño de la población permanece constante.
- **Función fitness (de ajuste):** Se trata de una función evaluadora de la calidad de un individuo como solución al problema. Permite la ordenación de los individuos de la población en cuanto a la bondad de los mismos.
- **Cruce:** Es una de las operaciones fundamentales que intervienen en todo algoritmo genético. Como norma general se aplica después de un proceso de selección de dos individuos y consiste en una combinación de los mismos para obtener como resultado otros dos nuevos individuos.
- **Mutación:** Es también una operación fundamental en un algoritmo genético. En este caso se selecciona un individuo, el cual sufre una pequeña modificación aleatoria en su codificación obteniéndose otro individuo nuevo.

En líneas generales y teniendo en cuenta la estructura de un algoritmo evolutivo, la estrategia operativa de un algoritmo genético consiste en partir de una población inicial de individuos generada aleatoriamente, cada uno de los cuales representa una posible solución del problema. Estos individuos se evalúan mediante una función *(fitness)*, que indica la calidad de la solución o grado de adaptación del individuo al entorno. A partir

de esta situación inicial se realizan una serie de iteraciones en cada una de las cuales se simula la creación de una nueva generación de individuos a partir de la generación anterior. Este proceso consiste en aplicar los operadores genéticos de selección, cruce y mutación sobre los individuos hasta llegar finalmente a una población que, si el algoritmo converge adecuadamente, estará compuesta por buenos individuos, siendo el mejor de estos la solución que ofrece el algoritmo. Un proceso de los llamados elitistas siempre mantendrá en la población al mejor individuo alcanzado hasta el momento.

Un algoritmo genético realiza una búsqueda multidireccional. Esta búsqueda es más intensiva en las direcciones más prometedoras, pero se dedica también una parte del esfuerzo a considerar regiones en principio de menor interés. En un intento de evitar caer en mínimos locales, que es el mayor problema de los algoritmos de búsqueda en Inteligencia Artificial, la población sufre una evolución simulada: en cada generación los individuos buenos se reproducen con mayor probabilidad que los malos; en este proceso se generan individuos nuevos que heredan algunas características de sus progenitores, mientras que otras son propias de cada individuo. Así pues se puede afirmar que los algoritmos genéticos tratan de alcanzar un equilibrio entre dos objetivos aparentemente en conflicto: explotar las buenas cualidades de las potenciales soluciones y explorar el espacio de búsqueda.

La Programación genética (*Genetic Programming*[3]) es una extensión de la técnica de algoritmos genéticos, originada a partir del trabajo de Koza (Koza 1992). Se incluye dentro del conjunto de métodos conocidos como *computación evolutiva* (Bäck et al. 1997). GP se ha usado en campos tales como la construcción automática de circuitos, identificación de sistemas no lineales en ingeniería química (identificación de variables relevantes), regresión simbólica, etc. (McKay et al. 1996, McKay et al. 1997, Willis et al. 1997). En la estimación del coste del software, GP se utiliza como una regresión simbólica automática para obtener las ecuaciones. La idea consiste en generar de forma aleatoria un conjunto inicial de ecuaciones que relacionen las variables de entrada y de

[3]En adelante GP.

salida, y seleccionar las ecuaciones de acuerdo al principio de que sobreviva el mejor. El proceso se repite hasta que se alcanza una solución aceptable. Se trata, por lo tanto de un método no paramétrico. La forma de una ecuación generada aleatoriamente podría ser:

$$Esfuerzo = 999,9 \cdot loc - 22222 \cdot loc^2$$

El algoritmo de Dolado (Dolado & Fernández 1998, Dolado et al. 1998) que ofrece una solución para regresión simbólica es el mostrado a continuación:

```
1. Generar la población inicial de ecuaciones
2. Mientras haya generaciones que procesar hacer
     2.1. Evaluar el fitness (adecuación) de cada ecuación
     2.2. Para cada ecuación de la población seleccionar de forma
     aleatoria uno de los operadores
          a) mutación con probabilidad Pm
          b) cruce con probabilidad Pc
          c) paso directo a la siguiente generación con probabilidad
          (1-Pm-Pc)
     2.3. Añadir la nueva ecuación a la nueva población
     Fin del Para cada
Fin del Mientras
```

Cada ecuación de la población se representa en forma de árbol. La operación de cruce toma un subárbol de cada una de las dos ecuaciones involucradas, y los intercambia, formando dos nuevas ecuaciones. La mutación, altera de forma aleatoria los subárboles de una ecuación.

Para más información sobre programación genética aplicada a estimación del coste consultar los trabajos de Dolado (Dolado 2001), (Dolado & Fernández 1998), (Dolado 1998), (Dolado et al. 1998), en los que se compara la utilización de GP, redes neuronales y regresión lineal en la estimación de esfuerzo. Una de las conclusiones principales es que no hay ningún método que destaque sobre el otro, y que GP se puede usar como alternativa o como complemento a la regresión lineal. En Shan (Shan et al. 2002) se aplica un método de computación evolutiva (GGGP *Grammar Guided Genetic Programming*)

en la estimación de esfuerzo, obteniendo significativamente mejores resultados que utilizando regresión lineal simple. Por último en Aguilar-Ruiz (Aguilar-Ruiz et al. 2001) se combina el uso de los Simuladores de Proyectos de Software (SPS *Software Project Simulators*) y la Computación Evolutiva, probándose la gran potencia de ésta última.

Inducción de Reglas

Otro de los métodos de Machine Learning, que se usa en la construcción de modelos de coste del software, es el de Inducción de Reglas (*Rule Induction*[4]). Se trata de una forma particular de aprendizaje inductivo en el cual el algoritmo produce reglas como resultado del proceso de modelización. RI se basa en:

> "...los algoritmos de inducción que reciben un conjunto de ejemplos de entrenamiento, cada uno de los cuales se describe por los valores atributo/resultado, construyen de forma automática árboles de decisión, que son capaces de clasificar de forma satisfactoria no solamente los ejemplos de entrenamiento, sino también los casos nuevos que se pueden presentar."(Kennedy et al. 1997), página 147.

El aprendizaje inductivo consiste en la adquisición de conceptos generales a partir de ejemplos específicos. Mediante el análisis de muchos ejemplos, es posible derivar un concepto general que establezca las características que definen esos ejemplos. Para producir un conjunto de reglas, la inducción opera sobre un conjunto de ejemplos de entrenamiento. Las reglas obtenidas se validan mediante un conjunto de ejemplos de test.

RI puede ser usado en un gran rango de problemas, siempre que exista un conjunto numeroso y adecuado de ejemplos. Las reglas se pueden ver como un árbol de decisión, donde las hojas contienen el valor o rango de valores predicho. Los árboles de decisión numéricos se generan calculando el resultado medio para cada conjunto de casos considerados en cada nodo. Como ejemplo se puede a continuación un fragmento de reglas generadas a partir del conjunto de datos de Desharnais (Desharnais 1989).

[4]En adelante RI.

```
If PFAjus ≥ 266 and
     If ExpAP < 3 and
          Transacciones < 165
     Then Esfuerzo = 3542
```

Una ventaja del aprendizaje inductivo sobre las redes neuronales es que las reglas son transparentes, y por lo tanto pueden leerse y entenderse, permitiendo un análisis posterior, cosa que no ocurre con las redes neuronales. En el fragmento anterior, se observa como se tienen en consideración, en primer lugar, los puntos de función ajustados, en segundo lugar la experiencia del administrador de proyecto, y por último las transacciones procesadas. Los defensores de las técnicas de estimación basadas en RI argumentan que la disponibilidad de las reglas ayudan al estimador a entender la predicción realizada y cualquier suposición subyacente sobre la que se base ésta. El estimador ve como se ha realizado la estimación y así la entiende mejor, e incluso es capaz de rechazarla dado el caso.

En el trabajo de Mair (Mair et al. 2000), se comparan redes neuronales, razonamiento basado en casos e inducción de reglas para predecir el esfuerzo software. Los sistemas se comparan en términos de exactitud, grado de claridad y configurabilidad. Las redes neuronales son más exactas que las reglas inducidas, sin embargo son menos transparentes y más difíciles de configurar que estas últimas. Otros trabajos sobre el tema pueden ser por ejemplo los de Gray, Jorgensen y Shepperd (Gray & McDonell 1997), (Jorgensen 1995), y (Shepperd & Kadoda 2001).

CART - Classification and Regression Trees. Árboles de Clasificación y de Regresión

Ambos se basan en el mismo principio, pero tienen diferentes objetivos. Los árboles de regresión se usan cuando el valor de salida a predecir es un valor continuo, mientras que los árboles de clasificación (también conocidos como árboles de decisión) se usan para predecir valores categóricos (ordinales o nominales). Esto hace a los árboles de regresión apropiados para usar en la estimación del coste del software. Un árbol de regresión, clasifica instancias (proyectos de software) con respecto a una cierta variable (esfuerzo, o productividad, por ejemplo). Consiste en un conjunto de reglas de la forma:

if (condición 1 y condición 2 y ...) then Z

visualizadas en forma de árbol binario, donde la variable dependiente Z es por ejemplo el esfuerzo. Cada nodo en el árbol de regresión, especifica una condición basada en una de las variables del proyecto que tiene influencia sobre la variable dependiente Z. Cada rama se corresponde con un valor posible para esa variable. La construcción de un árbol de regresión lleva asociada la división recursiva del conjunto de datos, hasta que se alcanza un criterio de parada.

Algunos trabajos sobre CART son los de Briand (Briand et al. 1999), en el que se comparan *stepwise regression*, *stepwise ANOVA*, CART, y un método basado en analogía utilizando el algoritmo ANGEL. El análisis de regresión, y los árboles de regresión produjeron mejores resultados que los restantes métodos.

Briand (Briand et al. 2000) realiza el mismo estudio que se describe en el apartado anterior, pero sobre datos de la Agencia Espacial Europea. Los árboles de regresión no se comportaron de la misma forma.

Kitchenham (Kitchenham 1998) compara *stepwise ANOVA* y CART. Los resultados obtenidos, utilizando los datos COCOMO, demostraron la sensible superioridad en valores MMRE de ANOVA frente a CART.

Para más información sobre estas técnicas se puede consultar Breiman y Wieczorek (Breiman et al. 1993) y (Wieczorek & Briand 2001).

La lista de trabajos que tratan técnicas de Inteligencia Artificial y combinaciones de ellas, para la estimación del coste del software es interminable, ya que son cada vez mayores los intentos para mejorar el complicado mundo de la estimación del coste. Aquí solo se han examinado algunos de ellos para dar una idea del panorama actual. Sin embargo existe otro método de machine learning que no se ha utilizado hasta ahora en la estimación del esfuerzo software, se trata de las técnicas de clustering. En el trabajo de Lee (Lee et al. 1998) se utiliza clustering, pero para ayudar a configurar la red neuronal que se utilizará para realizar las estimaciones. En el presente trabajo se utilizará el clustering en otro sentido. En la sección 2.4 se presenta el estado actual de los diferentes

algoritmos de clustering.

Clustering en Ingeniería del Software

En esta sección se verán los trabajos realizados en Ingeniería del Software, que utilizan algoritmos de clustering con diferentes propósitos. Se dejará para el final de este capítulo la clasificación de estos algoritmos de clustering, comentando más en profundidad esta técnica en próximos capítulos.

Anita Lee et. al (Lee et al. 1998), combina la utilización del análisis mediante técnicas de clustering con redes neuronales. El clustering se usa en una fase previa para facilitar, en una segunda fase la forma de la red neuronal, es decir, en cierta medida es una forma distinta de entrenamiento a backpropagation. De este modo se obtienen mejores resultados que si solamente se utiliza una red neuronal con backpropagation.

En el trabajo de Donald J. Reifer et. al (Reifer et al. 1999), se establece un mecanismo para convertir los ficheros COCOMO 81 con el fin de poder ser utilizados con el nuevo modelo de estimación del coste del sofware COCOMO II. Es en la fase de verificación de la exactitud del proceso, cuando los autores clusterizan en subconjuntos la base de datos de proyectos. Se tiene en cuenta la naturaleza dispar de los diferentes tipos de proyectos, con el fin de realizar la evaluación, obteniéndose un incremento en la precisión obtenida.

Serge Oligny et. al (Oligny et al. 2000), utilizan la base de datos ISBSG versión 4 para obtener un modelo empírico de predicción de la duración de tiempos, para realizar proyectos de ingeniería del software, basándose en el esfuerzo. Este modelo utiliza análisis de regresión para obtener la relación entre el tiempo y el esfuerzo. La novedad de este trabajo consiste en que no utiliza un único modelo para los 312 proyectos de la base de datos, sino que utiliza tres modelos, uno para los 208 proyectos desarrollados para *mainframe*, otro para los 65 proyectos desarrollados para *mid-range*, y por último otro para los 39 proyectos desarrollados para *pc*. Los autores diferencian entre las distintas plataformas de desarrollo utilizadas para realizar el modelado, obteniéndose una relación

distinta para cada uno de estos grupos.

En el trabajo de Erik Stensrud et. al (Stensrud et al. 2002), se realiza un estudio de las medidas de evaluación de los modelos de predicción del coste del software, MRE, MMRE, MdMRE y PRED(k). En concreto se examina la relación existente entre estas medidas y el tamaño del proyecto. Una de las conclusiones a las que se llega es que MRE no es independiente del tamaño del proyecto, de hecho, MRE es mayor para proyectos pequeños que para proyectos grandes, y viceversa. Una posible causa puede ser que los conjuntos de datos exhiben una cierta *heteroscedasticity*[5] (datos con distinta varianza). Por ello, estos autores recomiendan dividir los datos en dos conjuntos, uno con los proyectos de pequeño tamaño y otro con los de gran tamaño, aplicando a cada uno de ellos el criterio MMRE.

Xu & Khoshgoftaar (Xu & Khoshgoftaar 2004), proponen una técnica de estimación del coste basándose en una técnica de lógica difusa, que integra clustering difuso, proyección espacial, influencia difusa, y proceso de concreción (defuzzification). El algoritmo de clustering difuso que utilizan es *fuzzy k-medias*. Seleccionan un número de clusters a obtener y se aplica el algoritmo al conjunto de proyectos bajo estudio (COCOMO81). Para cada uno de los clusters obtenidos se obtienen un conjunto de reglas de inferencia, que se utilizan en el proceso de estimación. Los resultados dados por esta técnica, mejoran los obtenidos utilizando los tres modelos COCOMO81.

Una utilización, un tanto diferente, de un algoritmo de clustering, en concreto clustering jerárquico aglomerativo, es la que ofrecen Lung et. al (Lung et al. 2004). Estos autores desean obtener un diseño de software con un grado de cohesión y acoplamiento adecuados, mediante la aplicación de este tipo de algoritmos a unas especificaciones de requisitos software iniciales.

Como se aprecia en la explicación realizada de los trabajos expuestos en esta sección,

[5]Se podría traducir como *heterocedástico* pero se usará el término inglés.

el de Oligny (Oligny et al. 2000) es el que más ha servido como inspiración para esta tesis, la cual es original, ya que utiliza un algoritmo de clustering para segmentar toda la base de datos de proyectos, de forma automática y en base a ciertos atributos seleccionados manualmente, y aplica a cada uno de los segmentos obtenidos un modelo paramétrico de estimación. Siendo esto diferente a todo lo realizado con anterioridad.

2.3. Modelos de Estimación del Coste del Software

A continuación se muestran en orden cronológico de aparición y en más o menos detalle, muchos de los modelos que se utilizan o se han utilizado en la estimación del coste del software. La mayoría de ellos ya han sido mencionados en las diferentes clasificaciones vistas en los apartados anteriores, siendo las citas correspondientes (Boehm 1981), (DeMarco 1982), (Conte et al. 1986), (Kitchenham et al. 2002), (Fairley 1992), (Walkerden & Jeffery 1997), (Boehm, Abts & Chulani 2000) y (Wieczorek & Briand 2001).

2.3.1. SDC - System Development Corporation (1965-1967)

En 1964, la USAF (*United States Air Force*) junto con la empresa SDC, trabajaron juntas para estimar el coste de un proyecto. El propósito era obtener una serie de ecuaciones que permitieran estimar el coste de diseño, codificación y prueba de un programa. En 1967 ya habían recopilado 104 atributos de 169 proyectos de software completados. Mediante análisis estadístico de los atributos se determinaron de entre éstos, los 14 cost drivers más significativos:

1. Falta de requisitos
2. Estabilidad del diseño
3. Porcentaje de instrucciones matemáticas
4. Porcentaje de instrucciones de almacenamiento y recuperación
5. Número de subprogramas

6. Lenguaje de programación
7. Aplicación de gestión
8. Programa unimodular
9. Primer programa en el ordenador
10. Desarrollo concurrente del software
11. Uso de dispositivos de acceso aleatorio
12. Hardware anfitrión diferente
13. Número de errores personales
14. Desarrollado para una organización militar

La forma de la ecuación para estimar el coste total, en hombres-mes, es:

$$e = \sum_{i=0}^{14} a_i x_i$$

donde las x_i representan cada uno de los atributos anteriormente mencionados, y las a_i son constantes a determinar por el estimador.

Las fuentes utilizadas en la recopilación de esta información han sido las de Boehm y Nelson (Boehm 1984, Nelson 1967).

2.3.2. Modelo de "Farr and Zagorski" 1965

Este modelo los autores (Farr & Zagorski 1965) definen 13 factores de productividad, los cuales influyen sobre la productividad del programador. Estos 13 factores se determinaron mediante análisis de regresión de los datos de coste. Los factores dominantes son el número de instrucciones entregadas, documentos de uso interno y externo, y el número de palabras de la base de datos. Para estimar el esfuerzo en hombres-mes (men-month), se desarrollaron tres ecuaciones lineales similares a las del modelo SDC.

2.3.3. Aron (1969)

J. D. Aron, de IBM, estudió varios métodos para estimar el coste del software (Aron 1969). Aron observó que el equipo de desarrollo de un proyecto de software se va incrementando gradualmente hasta que alcanza un máximo, que se encuentra aproximadamente en la fase de pruebas, y luego disminuye hasta la finalización del producto.

Aron propuso varios métodos para estimar los costes del software:

- *Método basado en la experiencia.* Se basa en la experiencia obtenida al trabajar con proyectos similares.

- *Método basado en restricciones.* Es equivalente a una suposición bien fundada, donde el administrador está de acuerdo en la realización del proyecto con unas restricciones dadas.

- *Unidades de trabajo.* El proyecto total se divide en unidades de trabajo más pequeñas y manejables, para las que se establece el coste en llevarlas a cabo en base a la experiencia en la realización de unidades similares.

- *Estimación cuantitativa.* El proyecto total se divide en tareas más pequeñas, las cuales se clasifican en sencillas, medias y complicadas, basándose en el número de interrelaciones con otras tareas. Los factores de productividad (instrucciones por hombres-mes) se establecen en 500, 250 y 125 para tareas sencillas, medias y complicadas, respectivamente. El esfuerzo para cada tarea se calcula dividiendo el número de instrucciones entre la productividad apropiada. El esfuerzo total es la suma del esfuerzo de las distintas tareas que componen el proyecto:

$$e = \sum_{i=1}^{n} \frac{s_i}{p_i}$$

  donde e representa el esfuerzo medido en hombres-mes, s_i el tamaño en número de instrucciones de la tarea i, y p_i la productividad media de esa tarea teniendo en cuenta el tipo al que pertenece.

2.3.4. Modelo NAVAIR - NAVal AIR Development Center Model (1971)

Este modelo se usó para predecir el coste total en investigación, desarrollo, test y evaluación de nuevos sistemas de aviónica (Buck 1971). Utiliza una ecuación lineal parecida a la segunda ecuación del model de Farr-Zagorski para estimar el esfuerzo requerido en hombres-mes. La diferencia principal es que el esfuerzo en documentación se triplica. Los factores de coste más importantes son el número de instrucciones entregadas, tipos de documento, número de consolas independientes, experiencia media de los programadores y porcentaje de nuevas instrucciones (Mohanty 1981).

2.3.5. PRICE H - RCA (1973)

Se trata de una herramienta de estimación del coste hardware desarrollada en los años 60 por Frank Freiman en RCA. Estuvo disponible comercialmente a partir de 1973. PRICE H fue la primera herramienta de estimación del coste implementada en ordenador, y se extendió para manejar la estimación de software en 1979 (PRICE S).

2.3.6. TRW Wolverton TRW (1974)

Ray W. Wolverton, trabajando en el TRW Systems Group, desarrolló un algoritmo de estimación del coste basado en la suposición de que el coste varía proporcionalmente con el número de instrucciones (Wolverton 1974). Se trata de un algoritmo de cinco pasos, siendo el primero de ellos diferenciar el tipo de software dentro de seis categorías:

1. Rutinas de control
2. Rutinas de entrada/salida
3. Algoritmo pre o post-procesador
4. Algoritmo que realiza operaciones lógicas o matemáticas
5. Rutinas de manejo de datos
6. Aplicaciones de tiempo real

El segundo paso consiste en la estimación de tamaño y complejidad. Cada rutina se clasifica en nueva o vieja, y si su desarrollo será fácil, medio o complicado.

El tercer paso consiste en identificar las fases de desarrollo y test. En el cuarto paso se definen las actividades de cada fase y por último, en el quinto paso se establece el calendario de realización.

2.3.7. Tecolote - Tecolote Research, Inc. (1974)

Este modelo realiza la estimación de software para proyectos que requieren alta fiabilidad, por ejemplo sistemas de control de incendios. El modelo incluye diversas arquitecturas (una única CPU, múltiples CPU's, etc.), pero se basa principalmente en el tamaño estimado del programa (en instrucciones máquina), velocidad del computador y capacidad de la memoria (Frederic 1974).

2.3.8. GRC - General Research Corporation (1974-79)

El modelo GRC calcula el coste del software C_s (en dólares) como una función no lineal del número de instrucciones (Taback & Ditimore 1974).

$$C_s = 0,232 \cdot I^{1,43}$$

2.3.9. ESD - Electronic Systems Division 1975-78

Es similar al modelo Wolverton, excepto en que el número de líneas de código se refiere a las líneas fuente más que a instrucciones del lenguaje máquina. Una instrucción fuente equivale de entre una a tres instrucciones máquina. El coste del software se obtiene multiplicando el coste por instrucción y el número de instrucciones. Además se añade el coste debido al tipo de programa desarrollado (Mohanty 1981).

2.3.10. Doty (1977)

El modelo Doty (Herd et al. 1977) es el resultado de una intensa actividad de análisis de datos. Se desarrollaron una serie de ecuaciones de forma similar pero para diferentes tipos de aplicación. Por ejemplo, la ecuación para una aplicación de propósito general

viene representada por:

$$MM = 5,288 \cdot (KDSI)^{1,047} \text{ para } KDSI \geq 10$$

$$MM = 2,060 \cdot (KDSI)^{1,047} \left(\prod_{j=1}^{14} f_j \right) \text{ para } KDSI < 10$$

donde KDSI son miles de instrucciones fuente y MM representa el esfuerzo en hombres-mes.

2.3.11. Aerospace (1977)

Modelo que estima de forma separada el esfuerzo para los programas de tiempo real y el resto de proyectos (James 1977).

$$\text{Tiempo Real: } MM = 0,057 \cdot I^{0,94}$$

$$\text{Resto de programas: } MM = 2,012 \cdot I^{0,404}$$

donde MM es el esfuerzo en hombres-mes, e I el número de instrucciones fuente.

2.3.12. Boeing (1977)

Se trata de un modelo algorítmico tabular, donde algunos de los cost drivers son: instrucciones fuente, tipo de programa y complejidad del programa (Black et al. 1977).

2.3.13. Walston y Felix / IBM-FSD (1977)

En 1972 Walston y Felix comenzaron un proyecto de medidas software en la *Federal Systems Division* de IBM, con el propósito de valorar los efectos del uso de la programación estructurada sobre el proceso de desarrollo software. Se recogieron datos de 60 proyectos, se trazaron las relaciones entre las líneas de código desarrolladas y el esfuerzo, y mediante regresión de mínimos cuadrados se obtuvo la siguiente ecuación (Walston & Felix 1977):

$$E = 5,2 \cdot L^{0,91}$$

E es el esfuerzo en meses y L son cientos de líneas de código fuente.

2.3.14. SLICE - System Life Cycle Estimation (1977)

Este modelo fue desarrollado por A. L. Kustanowitz en 1977 (Kustanowitz 1977). Se fundamenta en el desarrollo de un perfil del proyecto, determinando las fases más importantes del proceso de desarrollo y asignando un porcentaje del esfuerzo total a cada una de esas fases. A continuación se determinan los rangos (valor máximo y mínimo) de productividad esperada en MM basándose en el tipo de aplicación desarrollada, la experiencia, el entorno de desarrollo y el lenguaje de programación utilizado en cada una de esas fases. Por último se estima el número total de líneas de código fuente que tendrá el proyecto. Con estos datos se aplican las ecuaciones de cálculo. El esfuerzo total se obtiene añadiendo los costes de gestión y de documentación.

$$MM = \sum_{i=1}^{n} \frac{s_i}{p_i}$$

donde MM es el esfuerzo en hombres-mes, s_i el tamaño en líneas de código fuente, y p_i la productividad, ambos para los módulos de tipo i.

2.3.15. Modelo de Putnam (1978)

Larry Putnam basó su modelo (Putnam 1978) en el trabajo de Norden y Aron. Tras diferentes estudios, Putnam observó que la relación entre el esfuerzo y el tiempo se podía expresar como:

$$S_s = C_k \cdot K^{1/3} \cdot t_d^{4/3}$$

donde S_s es el número de instrucciones fuente, K es el esfuerzo del ciclo de vida en hombres año, t_d es el tiempo de desarrollo en años, y C_k es una *constante tecnológica* (una medida del grado tecnológico que se aplica al proyecto en estudio).

Conte (Conte et al. 1986) critica este modelo, ya que diversas propiedades controvertidas del mismo lo hacen poco exacto. Aún así, este modelo fue la base de SLIM, que fue la segunda herramienta de estimación del coste que salió al mercado. Actualmente SLIM se comercializa como SLIM-Estimate por la compañía de Larry Putnam, Quantitative Software Management[6].

[6] http://www.qsm.com.

2.3.16. Modelo COCOMO - COnstructive COst MOdel(1979)

El modelo COCOMO fue desarrollado por Barry Boehm en 1979 y descrito en su libro *Software Engineering Economics* (Boehm 1981), se le conoce como COCOMO 81 para diferenciarlo del modelo COCOMO II, desarrollado posteriormente. Se trata de uno de los modelos de coste del software más conocidos y mejor documentados, estando formado tanto por ecuaciones de estimación de esfuerzo como de estimación de tiempo. COCOMO es un conjunto de tres subsistemas de modelado:

- COCOMO básico
- COCOMO intermedio
- COCOMO detallado

Los modelos COCOMO intermedio y detallado se pueden clasificar como *corrected single-factor models* (ver 2.2.2), ya que sus estimaciones se ajustan mediante una serie de factores correctores multiplicativos. Estas correcciones indican el efecto de un proyecto específico sobre el esfuerzo causado por factores que influyen sobre el coste. COCOMO básico no usa tales correcciones, tratándose en este caso de un *single factor model*.

Todas las ecuaciones de esfuerzo de COCOMO tienen la misma forma:

$$E_{nom} = a \cdot S^b$$

donde E_{nom} es el esfuerzo nominal, medido en hombres-mes, y S es el tamaño medido en miles de instrucciones fuente. Los valores de los coeficientes a y b dependen del modelo COCOMO que se use (básico, intermedio, detallado) y del modo del proyecto (orgánico, semi-detached[7], embedded[8]). El modo del proyecto se refiere al tipo de entorno de desarrollo en el cual se produce el nuevo sistema y del tipo de software (por ejemplo la experiencia en el desarrollo de sistemas, necesidad de cumplimiento del software de los requisitos establecidos, desarrollo concurrente con nuevos productos hardware, etc.). En

[7]Se usa el término inglés.
[8]Se usa el término inglés.

la tabla 2.1 se hace una breve descripción de estos modos. En la tabla 2.2 se pueden ver los valores que toman los coeficientes a y b según el modelo COCOMO y el modo de proyecto tal y como se ha comentado anteriormente.

Modo	**Descripción**
Orgánico	Se han desarrollado aplicaciones similares en un entorno de desarrollo conocido. El proyecto actual no tiene restricciones severas ni requisitos.
Semidetached	El proyecto tiene unos requisitos y restricciones más rígidas que el submodelo orgánico pero aún son flexibles.
Embedded	Se trata de un proyecto nuevo del que no hay muchos antecendentes y que tiene restricciones económicas o temporales y requisitos severos.

Tabla 2.1: Modos de proyecto COCOMO

Modo	**Básico**		**Intermedio y Detallado**	
	a	b	a	b
Orgánico	2,4	1,05	3,2	1,05
Semidetached	3,0	1,12	3,0	1,12
Embedded	3,6	1,20	2,8	1,20

Tabla 2.2: Coeficientes de COCOMO

En el COCOMO Intermedio y Detallado, la estimación del esfuerzo se ve afectada por un factor de ajuste calculado a partir de la multiplicación de hasta 15 coeficientes correspondientes a cada uno de los cost drivers, que vienen a reflejar las características propias de cada proyecto. Cada uno de los factores se valora con un número real escogido dentro de una escala con cinco rangos en la que se sitúa al factor. Los rangos son Muy Bajo, Bajo, Nominal, Alto y Muy Alto. Un valor para un cost driver por debajo de uno indica que ese factor, en el rango en el que se sitúa en el proyecto que se está analizando, disminuye el esfuerzo necesario para realizarlo. Dichos cost drivers se estructuran en cuatro grupos:

1. Atributos de Producto. Tienen en cuenta las características y requisitos del proyecto que está siendo desarrollado.

2. Atributos de Ordenador. Se refieren a las limitaciones que tendría el producto software que está siendo desarrollado en función de la máquina y el sistema operativo sobre los que va a operar.

3. Atributos Personales. Hacen referencia a la aptitud del personal que va a trabajar en el proyecto.

4. Atributos de Proyecto. Se refieren a las características del entorno de desarrollo del proyecto.

En la tabla 2.3 se pueden observar estos factores de forma resumida.

Grupo		**Cost Driver**
Atributos de Producto:	RELY	Fiabilidad Requerida del Software
	DATA	Tamaño de la Base de Datos
	CPLX	Complejidad del Producto
Atributos de Computador:	TIME	Restricción de Tiempo de Ejecución
	STOR	Restricción Principal de Almacenamiento
	VIRT	Inestabilidad de la Máquina Virtual
	TURN	Tiempo en torno al Computador
Atributos Personales:	ACAP	Capacidad del Analista
	AEXP	Experiencia en Aplicaciones
	PCAP	Destreza en la Programación
	VEXP	Experiencia en la Máquina Virtual
	LEXP	Experiencia en el Lenguaje de Programación
Atributos del Proyecto:	MODP	Uso de prácticas modernas de programación
	TOOL	Uso de herramientas de software
	SCHED	Restricciones en la duración del proyecto

Tabla 2.3: Cost drivers de las formas Intermedia y Detallada de COCOMO

La mejora de las formas intermedia y detallada de COCOMO no reside solamente en la introducción de los cost drivers. COCOMO Intermedio se puede usar cuando se han identificado las principales componentes del producto software. Esto permite realizar la estimación del esfuerzo en base a componentes, usando el tamaño estimado y los

valores de los cost drivers de forma apropiada para cada componente. Las estimaciones ajustadas para cada componente se suman para obtener una estimación global. De esta manera la estimación del esfuerzo se hace de una forma elegante, no solamente por la introducción de los cost drivers sino también por el uso de conocimiento detallado de las características internas del producto. COCOMO detallado va un poco más allá, usando diferentes multiplicadores de los cost drivers para cada fase principal de desarrollo.

2.3.17. Modelo PRICE S - RCA (1979)

PRICE S (Programming Review of Information Costing and Evaluation - Software) fue desarrollado por Frank Freiman y Robert Park en RCA para uso interno. En 1977 fue lanzado como modelo propietario y usado por el DoD (Department of Defense) de EEUU, la NASA, y otros proyectos de software del gobierno. Las ecuaciones del modelo no se hicieron públicas. En 1979 se convirtió en la primera herramienta software de estimación del coste. Actualmente la distribuye PRICE Systems[9].
Este modelo está formado por tres submodelos que son capaces de estimar el coste, el calendario para el desarrollo, y el mantenimiento de sistemas informáticos. Los tres submodelos son:

1. Submodelo de Adquisición. Este submodelo predice el coste y el calendario. Cubre todos los tipos de aplicaciones software, incluyendo los sistemas de gestión, comunicaciones, control, aviónica, y sistemas espaciales. También tiene en cuenta aspectos modernos de la ingeniería del software como son la reingeniería, la generación automática de código, el desarrollo en espiral, el desarrollo rápido, el prototipado rápido, el desarrollo orientado a objetos y la medición de la productividad del software.

2. Submodelo de Estimación del Tamaño. Predice el tamaño del software que va a ser desarrollado. El tamaño puede ser obtenido en líneas de código fuente, puntos de función, y/o puntos objeto (object points).

3. Submodelo del Ciclo de Vida. Se usa para un rápido cálculo de coste de la fase

[9]http://www.pricesystems.com.

de mantenimiento del software. Se usa de forma conjunta con el Submodelo de Adquisición.

Las fuentes utilizadas en esta recopilación han sido las de Boehm y Heemstra (Boehm, Abts & Chulani 2000, Heemstra 1992).

2.3.18. FPA - Function Point Analysis (1979)

Desarrollado por Alan Albrecht y John Gaffney de IBM (Albrecht & Gaffney 1983) para estimar el tamaño y el esfuerzo de desarrollo para un sistema de información de gestión. Se crea como una alternativa a la estimación del tamaño de un producto software mediante líneas de código fuente (SLOC - Source Lines Of Code). Los puntos de función tienen mayor nivel de abstracción que las SLOC, teniendo en cuenta aspectos como el número de tipos de transacciones de entrada o el número de informes distintos generados por el sistema. Cuando fueron presentados, los puntos de función constituían un modelo completo de estimación de esfuerzos, y la ecuación que se recoge aquí corresponde a ese modelo. El método de estimación de puntos de función se puede utilizar para determinar el tamaño del software que va a ser desarrollado, el cual se utilizará como una variable de entrada para algunos otros modelos de estimación de esfuerzo.

Los puntos de función presentan algunas ventajas frente a las SLOC, así por ejemplo, se pueden estimar antes en el ciclo de vida, ya que solo se necesita tener el documento de definición de requisitos. Esto es muy interesante, ya que si se utilizan los puntos de función como entrada de un modelo de estimación de esfuerzo y tiempo de desarrollo, se podrían conocer los datos de entrada del modelo muy pronto y con una buena aproximación, con lo que el esfuerzo consumido antes de hacer la estimación inicial es mucho menor. Además, los puntos de función son independientes del lenguaje de codificación, evitando los problemas de diferencia de tamaño que existen según sea el lenguaje utilizado en la codificación.

El cálculo de los puntos de función se realiza en dos fases:

1. Clasificar las funcionalidades de usuario en su categoría y calcular los puntos de función no ajustados teniendo en cuenta el nivel de información procesado por cada función, que puede ser simple, medio y complejo. Para cada nivel de cada función, teniendo en cuenta su categoría, habrá un número natural correspondiente a los puntos de función asignables a esa función.

   Existen actualmente 5 categorías de función (en el primer artículo se definieron sólo cuatro):

   a) Entradas externas (IT)

   b) Salidas externas (OT)

   c) Archivos lógicos internos (FT)

   d) Archivos de interfaz externos (EI)

   e) Consultas externas (QT)

2. Ajustar los puntos de función teniendo en cuenta la complejidad de la aplicación. Existen 14 características de complejidad:

   a) Comunicaciones de datos (C1)

   b) Funciones distribuidas (C2)

   c) Ejecución (C3)

   d) Frecuencia de uso de la configuración (C4)

   e) Velocidad de transacción (C5)

   f) Entrada de datos on-line (C6)

   g) Eficiencia del usuario final (C7)

   h) Actualización on-line (C8)

   i) Procesamiento complejo (C9)

   j) Reutilización (C10)

   k) Fácil instalación (C11)

l) Fácil funcionamiento (C12)

m) Localización múltiple (C13)

n) Facilidad de cambio (C14)

cada una con un rango de variación:

a) No presente o sin influencia = 0

b) Influencia insignificante = 1

c) Influencia moderada = 2

d) Influencia media = 3

e) Influencia significativa = 4

f) Influencia decisiva = 5

El cálculo de los puntos de función ajustados puede variar en ±35 % del cálculo original de puntos de función.
Las ecuaciones son:

1. **Tamaño:** $s_f = s_i \cdot \left(0,65 + 0,01 \sum_{i=1}^{14} c_i\right)$

donde s_f representa el tamaño final en puntos de función una vez ajustados. s_i representa el tamaño inicial en puntos de función no ajustados, y c_i representa cada uno de los 14 factores de complejidad.

2. **Esfuerzo:** $e = -13,39 + 0,0545 \cdot s$

donde e representa el esfuerzo medido en hombres-mes y s el tamaño en puntos de función.

2.3.19. Putnam SLIM (1979)

SLIM procede de Software LIfecycle Management, y fue desarrollado por Larry Putnam y Ann Fitzsimmons en Quantitative Software Management (QSM) en 1979, empresa fundada por ellos mismos. Se trata de la segunda herramienta comercial de estimación del coste del software, y puede encontrarse en SLIM-Estimate by QSM[10].

[10]http://www.qsm.com.

Este modelo se fundamenta en el análisis de Putnam del ciclo de vida del software basándose en que la distribución del tamaño del equipo de desarrollo de un producto software frente al tiempo, sigue una distribución de Rayleigh, y está inspirado en los trabajos de Norden (Norden 1963) y de Aron (Aron 1969). Norden observó a través de la representación gráfica de las frecuencias de distribución del personal durante las fases de desarrollo y mantenimiento de muchos proyectos llevados a cabo en IBM, que las curvas se asemejaban bastante a las curvas de distribución de Rayleigh, ya que el 90 % del proyecto era completado en dos tercios del tiempo total, mientras que el 10 % restante necesitaba un tercio del tiempo restante para completarse. Aunque esta distribución encontrada era meramente empírica, Norden no encontró ningún fundamento teórico para la misma.

SLIM soporta los métodos de estimación de tamaño más extendidos, incluyendo las líneas de código fuente y los puntos de función. Puede predecir el tamaño del proyecto, el esfuerzo, el tiempo de desarrollo y la proporción de defectos. Así como determinar la asignación de recursos necesarios para completar cada fase del proyecto.

Las ecuaciones del modelo no han sido editadas para el dominio público, ya que se trata de un software comercial, aunque los algoritmos centrales del modelo fueron publicados por Putnam (Putnam & Myers 1992). Son los que se recogen aquí:

1. Tamaño: $s = c \cdot (E_d)^{1/3} \cdot (t_d)^{4/3}$

donde s representa el tamaño en SLOC, E_d el esfuerzo total necesario para completar el proyecto, seleccionado de una base de datos de proyectos anteriores, c es una constante del proyecto denominada por Putnam *Factor Tecnológico*, este factor refleja el efecto de diferentes cost drivers tales como las restricciones de hardware, la complejidad del programa, los niveles de experiencia del personal y el entorno de programación. t_d es el

tiempo de desarrollo total del proyecto.

2. Esfuerzo: $E(t) = E_d \cdot \left(1 - e^{-at^2}\right)$; $a = \dfrac{1}{2\,(t_d)^2}$

donde $E(t)$ es el esfuerzo que se ha consumido en hombres-mes para desarrollar el proyecto durante t meses, a es una constante del proyecto que determina la pendiente de la curva, se obtiene también de proyectos anteriores.

2.3.20. Modelo Jensen (1979)

Desarrollado en 1979 por Randall Jensen en Hughes Aircraft Co. (Jensen 1983). Usa una ecuación de escalado nominal, parecida a la de Putnam, y un conjunto de cost drivers muy parecidos a los usados en los modelos Doty y COCOMO, pero con diferentes valores de los multiplicadores de esfuerzo. Algunos de estos valores están en función de los valores de otros factores. El modelo Jensen usa las curvas de Rayleigh para las distribuciones de esfuerzo, pero con una relación esfuerzo/calendario más compensada que SLIM.

La ecuación básica es:

$$S_s = C_{te} \cdot t_d \cdot K^2$$

donde S_s es el tamaño, t_d es el tiempo de desarrollo y K es el esfuerzo total. C_{te} es la constante tecnológica, y está dada por la ecuación:

$$C_{te} = C_{tb} \cdot \prod_{i=1}^{1s} f_i$$

El modelo Jensen fue evolucionando, incorporando nuevos cost drivers tales como la efectividad en la gestión, hasta llegar al modelo Jensen II en 1995.

El modelo Jensen fue la base de una serie de herramientas comerciales de estimación, tales como CEI, JS1, JS2, JS3 (Jensen Systems 1, 2 y 3), y el producto GAI SEER-SEM (1989). El modelo Jensen II finalizó en 1995, y es la base de la herramienta SAGE, comercializada por Software Engineering Institute[11].

[11] http://www.seisage.com/sage.htm.

2.3.21. Sech-Square (Chi-Cuadrado) / Parr(1980)

El modelo Putnam asume que al principio de cualquier proyecto de software el ratio de trabajo es cero, crece rápidamente y a continuación tiendo a cero asintóticamente. Pero se sabe que ciertas tareas, tales como planificación de alto nivel, estudios de viabilidad, etc., no se comportan de esta forma. Al inicio de un proyecto ya existe un grupo de personas que se encarga de tareas de organización y soporte, siendo por tanto el ratio de trabajo distinto a cero en el momento de iniciar el proyecto.
El modelo Sech-Square (Parr 1980) es un intento de ajustar el modelo Putnam, manejando el ratio de trabajo mediante una curva chi-cuadrado como se muestra a continuación:

$$m(t) = \frac{1}{4} \cdot sec^2 \cdot \left(\frac{a \cdot t + c}{2} \right)$$

donde $m(t)$ es el nivel del personal en el instante t, y a y c son parámetros que determinan la forma de la curva.

El modelo Sech-Square y el de Putnam son muy parecidos, excepto en las consideraciones al inicio del trabajo, (Mohanty 1981, Rook 1990).

2.3.22. Bailey-Basili (1981)

El punto de partida de este modelo, procedente de la Universidad de Maryland, es el hecho de que los modelos de coste previos no eran transportables y solamente válidos dentro de la organización en la que se habían desarrollado. Y esto debido a diferencias en los datos obtenidos, tipos de proyectos y factores de entorno. En 1981 estos autores desarrollaron un proceso para generar modelos para cualquier tipo de organización (Bailey & Basili 1981). Su trabajo estuvo influenciado principalmente por Boehm y Walston-Felix. Recogieron datos de 18 proyectos de la NASA/Goddard Software Engineering Laboratory y obtuvieron la siguiente ecuación:

$$MM_{NOM} = 3,5 + 0,73 \cdot (KLOC)^{1,16}$$

siendo MM el esfuerzo en hombres-mes, y KLOC miles de líneas de código.

2.3.23. SOFTCOST - Deep Space Network (DSN)- (1981)

Robert C. Tausworthe extendió los trabajos de Boehm, Herd, Putnam, Walston-Felix y Wolverton, para desarrollar un modelo de estimación del coste para el *Jet Propulsion Laboratory* de la NASA, conocido como *Deep Space Network Estimation Model* (Tausworthe 1981) o SOFTCOST. Este modelo es parecido al modelo Doty y al COCOMO ya que también está formado por una ecuación de escala nominal de la forma $MM_{NOM} = c \cdot (KLOC)^x$ y un conjunto de factores de ajuste del esfuerzo. SOFTCOST usa una ecuación lineal, así como la curva de Rayleigh para determinar la distribución del esfuerzo. Sin embargo la relación entre el esfuerzo y el calendario está más compensada que en el modelo SLIM.
SOFTCOST fue extendido posteriormente por Donal Reifer para producir el modelo comercial basado en PC *SOFTCOST-R*, comercializado actualmente por Reifer Consultants, Inc[12].

2.3.24. COPMO (1983)

El modelo COPMO, descrito en el trabajo de Conte (Conte et al. 1986), fue desarrollado por Thebaut (Thebaut 1983) y modeliza las relaciones existentes entre el nivel del personal, calendario, esfuerzo y tamaño. COPMO usa una ecuación básica de esfuerzo de la siguiente forma:

$$E = a + b \cdot S + c \cdot (P_a)^d$$

donde E es el esfuerzo en hombres-mes, S es el tamaño total del producto software en miles de líneas de código, y P_a es el tamaño medio del equipo de programadores (nivel del personal).

2.3.25. ESTIMACS (1983)

El model ESTIMACS fue desarrollado por Howard Rubin en *Hunter College* y fue comercializado por *Management and Computer Services (MACS)* en 1983 (Rubin 1983). MACS fue adquirido posteriormente por Computer Associates, siendo comercializada la

[12]http://www.reifer.com.

herramienta ESTIMACS como CA-Estimacs. Como ESTIMACS es un modelo propietario, sus detalles internos así como sus ecuaciones, no están disponibles. El modelo Estimacs está formado por 9 módulos, de los cuales el de esfuerzo es el más extenso e importante. Para producir una estimación el modelo realiza 25 preguntas relacionadas con la organización del usuario, y con la complejidad y tamaño del software a desarrollar. Estimacs usa como medida de tamaño puntos de función en vez de líneas de código.

2.3.26. SEER-SEM (1989)

Herramienta comercial[13] basada en el modelo original de Jensen. Se trata de una herramienta sofisticada que soporta las metodologías top-down y bottom-up. Sus ecuaciones de modelado son propietarias, pero es conocida su forma paramétrica.

2.3.27. COCOMO II (1995)

Se trata de la segunda versión de COCOMO[14], y aunque tienen el mismo nombre están sometidos a grandes diferencias (Boehm et al. 1995).

Este modelo está compuesto por tres submodelos:

1. Composición de aplicaciones.

   Este submodelo es usado para estimar el esfuerzo y el tiempo de desarrollo en proyectos que utilizan herramientas CASE (Computer Aided Software Engineering - Ingeniería del Software Asistida por Ordenador) para desarrollo rápido de aplicaciones. Estos productos aunque son muy diversos, son lo suficientemente simples como para ser rápidamente construidos a partir de componentes interoperables. Componentes típicos son los constructores del GUI (Graphical User Interface - Interfaz Gráfico de Usuario), gestores de bases de datos o de objetos, procesos de transacción, etc., así como componentes de dominios específicos como paquetes de control médico o industrial.

   Este modelo utiliza como variable de medida del tamaño del producto los Puntos Objeto (Banker et al. 1994, Kauffman & Kumar 1993). Los puntos objeto son

[13] http://www.gaseer.com.
[14] http://sunset.usc.edu/COCOMOII.

básicamente una cuenta de las pantallas, los informes y los módulos, desarrollados en un lenguaje de tercera generación en la aplicación. Cada cuenta es ponderada mediante un factor de complejidad de tres niveles, simple, medio y complejo. Sólo existe un calibrado de este modelo (Kauffman & Kumar 1993) debido a la falta de datos.

2. Diseño Inicial.

   Este submodelo tiene en cuenta la exploración de diferentes arquitecturas del sistema y conceptos de operación. Normalmente no es suficiente para hacer estimaciones de precisión.

   Utiliza como tamaño del producto los Puntos de función o las Líneas de código, cuando están disponibles.

   Introduce un conjunto de 5 Factores de escala. A cada uno de los cuales se le asigna un valor que será un número real correspondiente al rango en el que se sitúe el factor para el proyecto que se está desarrollando. Hay seis rangos: Muy Bajo, Bajo, Nominal, Alto, Muy Alto y Extra Alto. Los factores de escala son los siguientes:

   a) Precedentes (PREC). Tiene en cuenta la experiencia de la organización en el desarrollo de ese tipo de aplicaciones.

   b) Flexibilidad en el desarrollo (FLEX). Tiene en cuenta la rigidez de los requisitos y de las restricciones. Junto con la anterior sustituyen a los modos de desarrollo de COCOMO 81.

   c) Arquitectura/Solución de riesgos (RESL). Tiene en cuenta las medidas tomadas para la eliminación de riesgos.

   d) Cohesión del equipo (TEAM). Refleja las dificultades de cohesión y sincronización de los implicados en el proyecto, debido a su diversa procedencia y objetivos. Éstos son: usuarios, clientes, desarrolladores, equipo de mantenimiento, etc.

   e) Madurez de procesos (PMAT). Este factor tiene en cuenta el nivel de madurez de procesos de la organización, para lo cual utiliza la escala de cinco niveles

del CMM (Capability Maturity Model) desarrollado por el SEI (Software Engineering Institute) de la universidad de Carnegie Mellon.

Además este submodelo utiliza 7 Multiplicadores de Esfuerzo a los que se les asigna un número real teniendo en cuenta el rango, de seis niveles (Muy Bajo, Bajo, Nominal, Alto, Muy Alto y Extra Alto), en que se encuentre el multiplicador para el proyecto estudiado. La tabla de valores de multiplicadores de esfuerzo se calibra uniendo los multiplicadores de esfuerzo calibrados para el modelo Post - Arquitectura (USC-CSE 1997). Dichos cost drivers son:

a) Capacidad del personal (PERS)

b) Fiabilidad y complejidad del producto (RELY)

c) Reutilización requerida (RUSE)

d) Dificultad de la plataforma (PDIF)

e) Experiencia del personal (PREX)

f) Facilidades (FCIL)

g) Calendario (SCED)

3. Post-Arquitectura.

   Este submodelo puede ser utilizado cuando se ha completado el diseño de alto nivel y se dispone de información detallada sobre el modelo y, como su nombre sugiere, la arquitectura del software está bien definida y establecida. Permite realizar estimaciones para el conjunto del ciclo de vida de desarrollo y es una extensión del modelo de Diseño Inicial. Este modelo es parecido en estructura y formulación al modelo Intermedio de COCOMO 81. Utiliza Puntos de Función y/o Líneas de Código Fuente como entrada para el parámetro del tamaño del producto. Usa un conjunto de cinco factores de escala, iguales en su fundamento conceptual a los vistos anteriormente para el submodelo de diseño inicial, y diecisiete multiplicadores de esfuerzo, cada uno con un rango de seis niveles igual al visto para el submodelo de diseño inicial. Este submodelo ha sido calibrado sobre una base de datos de 161

proyectos recopilados de la industria comercial, aeroespacial, gubernamental y de organizaciones sin ánimo de lucro; utilizando una aproximación bayesiana (Chulani et al. 1998). Los cost drivers se agrupan en las siguientes cuatro categorías:

a) Producto
   1. Fiabilidad requerida del software (RELY)
   2. Tamaño de la base de datos (DATA)
   3. Complejidad del producto (CPLX)
   4. Reutilización requerida (RUSE)
   5. Documentación desarrollada (DOCU)

b) Plataforma
   6. Restricciones en el tiempo de ejecución (TIME)
   7. Restricciones en la memoria principal (STOR)
   8. Volatilidad de la plataforma (PVOL)

c) Personal
   9. Aptitud de los analistas (ACAP)
   10. Aptitud de los programadores (PCAP)
   11. Experiencia en el desarrollo de aplicaciones similares (AEXP)
   12. Experiencia con la plataforma de desarrollo (PEXP)
   13. Experiencia con el lenguaje y la herramienta (LEXP)
   14. Continuidad del personal (PCON)

d) Proyecto
   15. Utilización de herramientas software (TOOL)
   16. Desarrollo en múltiples localizaciones (SITE)
   17. Tiempo necesario para el desarrollo (SCED)

Las ecuaciones del modelo son las siguientes:

1. Esfuerzo: $e = a \cdot s^{\left(b+c\sum_{i=1}^{5} y_i\right)} \cdot \left(\prod_{j=1}^{n} x_j\right)$

donde e representa el Esfuerzo medido en hombres-mes, s el tamaño en SLOC, a, b y c son constantes, y x_j el valor del cost driver j, n vale 7 en el submodelo diseño preliminar y 17 en post-arquitectura, siendo y_i el factor de escala i.

2. Tiempo de desarrollo: $t = 3 \cdot (e)^{0,33+0,2\cdot(b-1,01)} \cdot \left(\frac{SCED\,\%}{100}\right)$

donde t representa el tiempo de duración total del desarrollo del proyecto en meses, e es el esfuerzo medido en hombres-mes, b es una constante y $SCED$ es el multiplicador de esfuerzo *"Tiempo necesario para el desarrollo"*.
Para consultar los valores de los coeficientes COCOMO II calibrados, ver Chulani (Chulani et al. 1998).

Existen varias herramientas basadas en este modelo:

1. COCOMO II. 1999.0 Desarrollada por el equipo de la Universidad del Sur de California[15] que ha automatizado el modelo teórico. Es gratuita.

2. COSTAR. Desarrollada por Costar[16]. Es una herramienta interactiva que permite realizar seguimientos de proyectos, así como realizar experimentos de simulación del tipo ¿qué sucedería si ...?

2.3.28. SPR Checkpoint (1996)

Checkpoint es una herramienta de estimación de proyectos de software (que se ejecuta sobre el sistema operativo MS-DOS) de *Software Productivity Research* (SPR), y desarrollada por Capers Jones (Jones 1996). Las ecuaciones de estimación han sido inferidas a partir de una base de datos propietaria de 8000 proyectos de software, centrando su atención en cuatro áreas que necesitan gestionarse para mejorar la calidad software y la productividad. Checkpoint usa puntos de función como variable principal de tamaño. SPR Checkpoint tuvo su continuación en SPR KnowledgePLAN (software de estimación que corre sobre sistemas Windows) perteneciente a Software Productivity Research[17].

[15] ftp://ftp.usc.edu/pub/soft_engineering/COCOMOII/cocomo99.0/c990windows.exe/.
[16] http://www.softstarsystems.com.
[17] http://www.spr.com.

2.3.29. COBRA (1998)

Se trata de un modelo de estimación híbrido, que combina aspectos algorítmicos y experimentales (Briand et al. 1998). El núcleo de COBRA está formado por dos componentes. El primer componente es un modelo que produce una sobre-estimación del coste. Y el segundo componente es un modelo de productividad que estima la productividad a partir de esa sobre-estimación.

2.3.30. SELECT (1998)

SELECT procede del nombre de la empresa que la ha desarrollado *The Object Factory* SELECT Software Tools[18] (SELECT Software Tools 1998).

Este modelo ha sido diseñado para sistemas distribuidos de gran tamaño. Está orientado a objetos y fundamenta sus estimaciones en componentes. Utiliza un ciclo de vida de desarrollo incremental, pero puede ser adaptado para otros modos de desarrollo. La naturaleza de sus entradas permite al modelo ser usado en cualquier etapa del ciclo de vida, lo que es muy interesante en la etapa de planificación cuando se tiene poca información del proyecto. En las etapas posteriores, cuanta más información está disponible, sus estimaciones se hacen más fiables.

La técnica real de estimación está basada en ObjectMetrix. Esta técnica trabaja midiendo el tamaño de un proyecto contando y clasificando los elementos del software dentro de un proyecto. Las técnicas de ObjectMetrix comienzan con una medida básica del esfuerzo en personas-día. Este esfuerzo supone que se llevan a cabo todas las actividades de un ciclo de vida del software normal, pero no tiene en cuenta nada sobre las características o de la tecnología involucrada que pudiera calificar las estimaciones de esfuerzo. Los perfiles de actividades predefinidas cubren planificación, análisis, diseño, programación, pruebas, integración y revisión, y se aplican de acuerdo al tipo de elementos del proyecto. Estos perfiles de actividad están basados sobre datos medidos, recopilados y mantenidos por *Object Factory*.

[18] http://www.theobjectfactory.com.

Una vez obtenido, el esfuerzo base es ajustado utilizando calificadores para añadir o quitar un porcentaje de la cantidad de esfuerzo estimado para cada actividad. Un factor de tecnología refleja el impacto de un entorno dado de programación, esta vez se añade o se sustrae un porcentaje a la cantidad estimada utilizando calificadores.

Aplicando los calificadores y el factor de tecnología al esfuerzo básico, calculado para cada elemento de proyecto, se produce una estimación del esfuerzo total en personas-día por actividad. Esta estimación total representa el esfuerzo requerido por una persona de un nivel de habilidad medio para completar el proyecto.

Utilizando el total estimado de esfuerzo para una persona, se determina el calendario en función del número de desarrolladores (entrada como variable independiente), los niveles individuales de habilidad, el número de días de trabajo productivos (excluyendo días perdidos en reuniones, enfermedad, etc.) por mes, y se considera un porcentaje adicional de contingencia.

SELECT Estimator adapta la técnica de estimación OBJECTMETRIX pero refinando los Calificadores y los Factores de Tecnología.

2.4. Clustering

El proceso de clustering consiste en la división de los datos en grupos de objetos similares. Para medir la similaridad entre objetos se suelen utilizar diferentes formas de distancia: distancia euclídea, distancia de Manhatan, distancia de Mahalanobis, etc. El representar los datos por una serie de clusters, conlleva la pérdida de detalles, pero consigue la simplificación de los mismos. El clustering es una técnica más de Machine Learning, en la que el aprendizaje realizado es no-supervisado (*unsupervised learning*). Desde un punto de vista práctico, el clustering juega un papel muy importante en aplicaciones

de data mining, tales como exploración de datos científicos (Cadez et al. 2001), recuperación de la información y minería de texto (Cutting et al. 1992, Steinbach et al. 2000, Dhillon et al. 2001), aplicaciones sobre bases de datos espaciales (tales como GIS o datos procedentes de astronomía) (Xu et al. 1998), aplicaciones Web (Cooley et al. 1999), marketing, diagnóstico médico, análisis de ADN en biología computacional (Ben-Dor & Yakhini 1999), y muchas otras.

Para obtener una introducción bastante buena a las técnicas de clustering se puede consultar Han y Kamber (Han & Kamber 2001). Se dispone en la bibliografía de una serie de trabajos clásicos que emplean o aplican clustering en la resolución de problemas en áreas concretas, algunos de ellos son como por ejemplo los de Arabie (Arabie & Hubert 1996) que usa el clustering en estadística, así como Massart en ciencia (Massart & Kaufman 1983). Una introducción clásica para reconocimiento de patrones (reconocimiento de voz y caracteres) se encuentra en el trabajo de Duda y Hart (Duda & Hart 1973). Así mismo, las técnicas de clustering se han aplicado también a la segmentación de imágenes y visión por computador (Jain & Flynn 1966). En Dempster (Dempster et al. 1977) se puede encontrar una aproximación estadística al reconocimiento de patrones. El clustering se puede ver como un problema de estimación de densidad, siendo éste el objeto de la estimación multivariable en estadística (Scott 1992). También se usa clustering para la compresión de datos en procesamiento de imágenes, también conocido como *vector quantization* (Gersho & Gray 1992). El análisis de datos mediante clustering también es tratado en Daniel y Wood (Daniel & Wood 1980).

Muchas investigaciones actuales en diferentes campos usan el clustering, tales como estadística, reconocimiento de patrones, y machine learning. En los últimos años ha surgido una gran variedad de algoritmos de clustering, y son los que serán clasificados y analizados a continuación (Berkhin 2002):

- Métodos Jerárquicos
    - Algoritmos Aglomerativos

    - Algoritmos Divisivos

- Métodos de Particionado y recolocación

    - Clustering Probabilístico
    - Métodos de los k-vecinos (*k-medoids*)
    - Métodos de las k-medias
    - Algoritmos basados en Densidad
        - Clustering de Conectividad Basada en Densidad (*Density-Based Connectivity Clustering*)
        - Clustering basado en Funciones de Densidad (*Density Functions Clustering*)

- Métodos basados en Rejillas
- Métodos basados en la Co-Ocurrencia de Datos Categóricos
- Clustering basado en Restricciones
- Algoritmos para Datos de Grandes Dimensiones

    - Clustering Subespacial
    - Técnicas de Co-Clustering

A continuación se presenta una breve descripción de las características de cada una de estas clasificaciones.

2.4.1. Clustering Jerárquico (*Hierarchical Clustering*)

Construye una jerarquía de clusters, o dicho de otra forma, un árbol de clusters, también conocido como *dendograma*. Cada nodo cluster contiene varios clusters hijo, y los clusters hermanos particionan los elementos que forman el cluster padre del que proceden. Tal aproximación permite explorar los datos en diferentes niveles de granularidad. Los métodos de clustering jerárquico se dividen en **aglomerativos** (bottom-up)

y **divisivos** (top-down) (Jain & Dubes 1988, Kaufman & Rousseeuw 1990). El clustering *aglomerativo* comienza haciendo de cada elemento un cluster, y de forma recursiva mezcla dos o más clusters (los más apropiados) para constituir uno nuevo. El clustering *divisivo* comienza con un solo cluster que contiene a todos los elementos, y recursivamente lo va dividiendo en los clusters más adecuados. Este proceso se desarrolla hasta que se alcanza un criterio de parada, normalmente alcanzar el número k de clusters deseado. La forma en la que se dividen o se agrupan los clusters depende de una medida de la similiridad o desimiliridad entre los elementos de dichos clusters. Los clusters contienen elementos similares entre sí.

Guha (Guha et al. 1998) introduce el algoritmo de clustering aglomerativo CURE (*Clustering Using REpresentatives*). Su principal característica es que representa un cluster por un número fijo c de elementos dispersos a su alrededor. La distancia usada entre dos clusters es la mínima de las distancias entre dos de los elementos representativos de cada cluster. CURE permite cubrir formas no esféricas.

Otro algoritmo jerárquico aglomerativo es CHAMELEON (Karypis et al. 1999), el cual utiliza modelado dinámico en la agregación de clusters. Este algoritmo encuentra clusters de diferentes formas, densidades y tamaños en el espacio bidimensional.

En el caso de clustering jerárquico divisivo se tiene el algoritmo PDDP (*Principal Direction Divisive Partitioning*) (Boley 1998). Este algoritmo bisecciona los datos en el espacio euclídeo mediante un hiperplano que pasa a través del centroide de los datos.

Otro algoritmo jerárquico a destacar, que no se puede clasificar dentro de las dos categorías ya conocidas, sería COBWEB (Fisher 1987). Se caracteriza porque utiliza aprendizaje incremental: en vez de dividir o agrupar los puntos, construye de forma dinámica un dendograma procesando los datos de uno en uno. Además COBWEB pertenece a los métodos de aprendizaje conceptual o basados en modelos. Esto significa que cada cluster se considera como un modelo que puede describirse intrinsicamente, más que un ente formado por una colección de puntos. El dendograma de COBWEB es un

árbol de clasificación.

2.4.2. Métodos de Particionado y Recolocación (*Partitioning Relocation Clustering*)

Estos algoritmos dividen los datos en varios subconjuntos. La comprobación de todos los posibles subconjuntos del sistema no es factible computacionalmente, por lo que es necesario la utilización de alguna técnica heurística avara para realizar una optimización iterativa. Esto se traduce en una reasignación (recolocación) iterativa de puntos entre los k clusters existentes, obteniéndose así y de forma gradual clusters más óptimos.

Clustering Probabilístico (*Probabilistic Clustering*)

En la aproximación probabilística, se considera que los datos bajo estudio siguen una mezcla de diferentes distribuciones de probabilidad (*mixture models*), (McLachlan & Basford 1988). El área alrededor de la media de cada distribución (supuestamente unimodal), constituye un cluster. De esta forma a cada cluster se le asocian los parámetros correspondientes de la distribución, tales como media, varianza, etc. El ajuste de los parámetros del modelo requiere alguna medida de su bondad, es decir, cómo de bien encajan los datos en la distribución que los representa. Este valor de bondad se conoce como el likelihood[19] de los datos. Se trataría entonces de estimar los parámetros buscados maximizando este likelihood. Normalmente, lo que se calcula es el logaritmo de este likelihood, conocido como log-likelihood[20] ya que es más fácil de calcular de forma analítica. La solución obtenida es la misma, gracias a la propiedad de monotonicidad del logaritmo. La forma de esta función log-likelihood es:

$$L(\Theta, \pi) = log \prod_{n=1}^{NI} P(x_n)$$

donde NI es el número de puntos, que suponemos independientes, Θ representa los parámetros buscados, y $P(x_n)$ es la función de densidad.

[19]Significa verosimilitud, y no será traducido al castellano.
[20]Término no traducido al castellano.

Un algoritmo que se ajusta a estos principios, y ampliamente utilizado es el algoritmo EM (*Expectation Maximization*). Este algoritmo será descrito en detalle en el capítulo dedicado a la evaluación del modelo paramétrico. Para una introducción rápida al algoritmo EM se pude consultar Mitchell (Mitchell 1997) y para más detalle Dempster y McLachlan (Dempster et al. 1977, McLachlan & Krishnan 1997).

Otro algoritmo muy utilizado es AUTOCLASS (Cheeseman & Stutz 1996), el cual utiliza también el modelo mixto, y cubre gran variedad de distribuciones, tales como la de Bernoulli, Poisson, Gaussiana y Log-Normal. AUTOCLASS, más que centrarse en un único modelo de mezcla de distribuciones, prueba entre diferentes modelos con diferentes distribuciones y diferente número de clusters k, quedándose finalmente con el óptimo.
El algoritmo MCLUST (Fraley & Raftery 1999) forma parte de un paquete software, conocido comercialmente como S-PLUS, y usa modelos Gaussianos con elipsoides de diferentes volúmenes, formas y orientaciones.

Por último destacar la bondad de estos métodos para realizar clustering sobre datos heterogéneos.

Método de los k-vecinos (*k-Medoids Methods*)

Se diferencia del método de las k-medias en la forma en la que se selecciona uno de los elementos como representante de cada cluster. Un vecino denota el dato más apropiado dentro de un cluster para representarlo. La utilización de este método tiene dos ventajas. La primera es que no existe ninguna limitación en su aplicabilidad a cualquier tipo de atributo, y la segunda es que la elección de los vecinos se realiza a partir una fracción predominante de puntos dentro del cluster (los más cercanos), no haciendo caso de puntos periféricos, por lo que es menos sensible a la aparición de outliers (valores atípicos).

Los clusters se definen como subconjuntos de puntos cercanos a sus respectivos vecinos, siendo definida la función objetivo como la distancia media u otra medida de

semejanza entre un punto y su vecino.

PAM (*Partitioning Around Medoids*) y CLARA (*Clustering LARge Applications*) (Kaufman & Rousseeuw 1990) son dos algoritmos basados en este método.

Método de las k-medias (*k-Means Methods*)

El método de las k-medias (Hartigan 1975, Hartigan & Wong 1979), es hasta ahora el más utilizado en aplicaciones científicas e industriales. El nombre le viene porque representa cada uno de los clusters por la media (o media ponderada) de sus puntos, es decir, por su centroide. Este método únicamente se puede aplicar a atributos numéricos, y los outliers le pueden afectar muy negativamente. Sin embargo, la representación mediante centroides tiene la ventaja de que tiene un significado gráfico y estadístico inmediato. La suma de las discrepancias entre un punto y su centroide, expresado a través de la distancia apropiada, se usa como función objetivo a minimizar.

La función objetivo, expresada como la suma de los cuadrados de los errores entre los puntos y sus centroides respectivos, es igual a la varianza total dentro del propio cluster. La suma de los cuadrados de los errores se puede racionalizar, como el negativo del log-likelihood, para modelos mixtos que utilicen distribuciones normales. Por lo tanto, el método de las k-medias se puede derivar a partir del marco probabilístico (para más información ver subsección *Clustering Probabilístico* de Mitchell (Mitchell 1997)).

Existen dos versiones del método de las k-medias. La primera es parecida al algoritmo EM, y se basa en dos pasos iterativos: primero reasigna todos los puntos a sus centroides más cercanos, y en segundo lugar recalcula los centroides de los nuevos grupos creados en el anterior. El proceso continua hasta alcanzar un criterio de parada (por ejemplo que no se realicen nuevas reasignaciones). Esta versión se conoce como algoritmo de Forgy (Forgy 1965).
La segunda versión (Duda & Hart 1973) reasigna los puntos basándose en un análisis más detallado de los efectos causados sobre la función objetivo al mover un punto de

su cluster a otro nuevo. Si el traslado es positivo, se realiza, en caso contrario se queda como está.

Algoritmos basados en Densidad (*Density Based Partitioning*)

Un espacio abierto en el espacio euclídeo se puede dividir en un conjunto de sus componentes conectadas entre sí. La implementación de esta idea (partición de un conjunto finito de puntos), requiere utilizar conceptos tales como densidad, conectividad y frontera. Un cluster definido como un componente denso conectado, crece en cualquier dirección que la densidad le permita. Por lo tanto, los algoritmos basados en densidad son capaces de descubrir clusters de formas arbitrarias, que no podrían ser descubiertas mediante por ejemplo, algoritmos de k-medias. Además esto le proporciona una protección natural a los outliers. Un inconveniente reside en el hecho de que un cluster denso está formado por dos áreas adyacentes con diferentes densidades, lo cual no ofrece demasiada información. Como los algoritmos basados en densidad requieren medidas espaciales, el campo de aplicación natural para ellos es el clustering de datos espaciales (Han & Kamber 2001).

***Density-Based Connectivity Clustering*:** Los conceptos fundamentales de este tipo de clustering, son densidad y conectividad, ambos medidos en términos de la distribución local de los vecinos más cercanos.

El algoritmo DBSCAN (Density Based Spatial Clustering of Applications with Noise) (Ester et al. 1996) es el más utilizado dentro de esta categoría, centrándose en los tipos de datos de baja dimensionalidad.

La pareja densidad-conectividad forman una relación simétrica, de manera que todos los puntos alcanzables desde los objetos centrales, pueden factorizarse en componentes maximalmente conectados formando clusters. Los puntos que no se conectan a ningún punto central, se declaran como outliers, y no son cubiertos por ningún cluster. Los puntos no centrales dentro de un cluster representan su frontera.

Otro algoritmo, GDBSCAN (Sander et al. 1998) introduce algunas generalizaciones sobre DBSCAN, que lo hacen más eficiente.
OPTICS (*Ordering Points To Identify the Clustering Structure*) (Ankerst et al. 1999), está también basado en el algoritmo DBSCAN, sobre el que realiza un ajuste de parámetros.
Mientras que OPTICS se puede considerar una extensión de DBSCAN, en el sentido de que utiliza densidades locales diferentes, una aproximación más matemática podría consistir en considerar una variable aleatoria igual a la distancia desde un punto a su vecino más cercano, y aprender su distribución de probabilidad. En vez de basarse en los parámetros definidos por el usuario, se podría suponer que cada cluster tiene su propia escala de distancia al vecino más cercano. La meta sería descubrir dichas escalas. Esta aproximación *no paramétrica* se implementa en el algoritmo DBCLASD (Distribution Based Clustering of Large Spatial Databases) (Xu et al. 1998).

***Density Functions Clustering*:** Hinneburg (Hinneburg & Keim 1998) en vez de centrarse en el cálculo de las densidades asociadas a los puntos de datos, fija su atención en el cálculo de funciones de densidad definidas sobre el espacio de atributos subyacente. Estos autores propusieron el algoritmo DENCLUE (DENsity-based CLUstEring). Junto con DBCLASD, se basa en firmes fundamentos matemáticos.

El algoritmo es estable frente a outliers. Además es escalable, ya que en su etapa inicial construye un mapa de cubos hiper-rectángulos. Por esta razón, el algoritmo se puede clasificar también como un método basado en Rejillas. Aplicaciones típicas de DENCLUE incluyen análisis de datos multimedia de alta dimensionalidad y datos procedentes de biología molecular.

2.4.3. Algoritmos basados en Rejillas (*Grid-Based Methods*)

En la sección 2.4.2 se han utilizado términos como densidad, conectividad, y frontera. Otra forma de tratar con estos términos, consiste en heredar la topología del espacio de atributos subyacente. Para limitar las múltiples combinaciones de búsqueda, solamente

se consideran segmentos multi-rectangulares. Un segmento (cubo, celda, región), es el producto cartesiano de subrangos (valores continuos en el caso de atributos numéricos) de valores de atributos individuales. Los métodos que particionan el espacio se suelen llamar métodos basados en rejillas. El segmento elemental se corresponde con valores individuales de subrangos, y se le conoce como unidad.

De esta forma, se cambia el enfoque desde atención a los datos, hacia el espacio particionado. El particionado de los datos se realiza a partir de la pertenencia de los puntos a segmentos que resultan del particionado espacial, el cual se basa en características de rejilla acumuladas de los datos de entrada. Una ventaja de estos métodos es que son independientes del orden de los datos, frente a los métodos de recolocación y todos los algoritmos incrementales, que son muy sensibles al orden de dichos datos. Los métodos de particionado basados en densidad trabajan bien con atributos numéricos, mientras que los métodos basados en rejillas se pueden aplicar sea cual sea el tipo de los atributos.

La metodología basada en rejillas refleja un punto de vista técnico. La categoría es ecléctica: el método contiene tanto algoritmos de particionado como jerárquicos. Por ejemplo, el algoritmo DENCLUE visto en la sección 2.4.2 usa rejillas en su etapa inicial.

Algunos algoritmos de este tipo son: BANG-clustering (Schikuta 1996), el cual mejora el algoritmo jerárquico (parecido a éste) GRIDCLUST (Schikuta & Erhart 1997); STING (STatistical INformation Grid-based method) (Wang et al. 1997), WaveCluster (Sheikholeslami et al. 1998), y FC (Fractal Clustering) (Barbara & Chen 2000).

2.4.4. Métodos basados en la Co-Ocurrencia de Datos Categóricos (*Co-Ocurrence of Categorical Data*)

En esta sección se tratarán los datos categóricos, que frecuentemente se relacionan con el concepto de una variable *transacción* (conjunto finito de elementos llamados *items*, pertenecientes a un universo común de items). Por ejemplo, los datos que representan el carro de la compra poseen esta forma. Cada transacción se pude presentar en un formato

punto-por-atributo, enumerando todos los items j, y asociando con una transacción los atributos binarios que indican si el item j-ésimo pertenece a la transacción o no. Esta representación es dispersa, de manera que dos transacciones aleatorias tienen muy pocos items en común. Ejemplos de casos que tienen características comunes a la representación punto-por-atributo para datos categóricos, serían datos de alta dimensionalidad con una cantidad significativa de valores cero, y una pequeña cantidad de valores comunes entre dos objetos. Los métodos convencionales de clustering, basados en medidas de similaridad, no trabajan bien en estos casos. Se han desarrollado diversos métodos de clustering basados en la idea de la *co-ocurrencia* de datos categóricos, y que se aplican en el análisis de perfiles de cliente, análisis web, etc.

El algoritmo ROCK (Robust Clustering algorithm for Categorical Data)(Guha et al. 1999) trata con datos categóricos y tiene muchas características en común con el algoritmo CURE (sección 2.4.1): se trata también de un clustering jerárquico, la aglomeración se lleva a cabo hasta que se ha construido un número específico k de clusters, y usa los datos de muestra en el mismo sentido que hace CURE.

El algoritmo SNN (Shared Nearest Neighbours) (Ertoz et al. 2002) combina una aproximación basada en densidad, con la idea de ROCK.
El algoritmo CACTUS (Clustering Categorial Data Using Summaries) (Ganti et al. 1999) busca clusters de forma hiper-rectangular (llamados *regiones de intervalo*), utilizando datos punto-por-atributo con atributos categóricos. Estos clusters son segmentos. CACTUS está basado en la idea de la co-ocurrencia para pares atributo-valor.
Otros algoritmos que encajan dentro de esta clasificación son HMETIS (Karypis et al. 1997), STIRR (Sieving Through Iterated Reinfircement) (Gibson et al. 1998), que trata con co-ocurrencias para objetos categóricos d-dimensionales, llamados *tuplas*.

2.4.5. Clustering basado e Restricciones (*Constraint-Based Clustering*)

En la vida real los problemas tienen restricciones diferentes. Los clusters se ven frecuentemente sometidos a limitaciones específicas del problema. Actualmente es objeto de estudio la realización de clusters sometidos a estas limitaciones (Han & Kamber 2001).

El marco de trabajo para el clustering basado en restricciones se introduce en la obra de Tung (Tung et al. 2001).

Una aplicación típica de estos métodos es el cluster de datos espaciales en dos dimensiones con presencia de obstáculos. En vez de la distancia euclídea, se usa la *distancia obstáculo*, que es una medida del camino disponible más corto entre dos puntos. El algoritmo COD (Clustering with Obstructed Distance) (Tung et al. 2001) recoge estas ideas. Un ejemplo de agrupación sería el de dos ciudades que está separadas por un puente, o que están unidas por un puente. En el primer caso se agruparían en dos ciudades diferentes, mientras que en el segundo caso se agruparían en una única ciudad.

2.4.6. Algoritmos para Datos de Grandes Dimensiones

Hay casos en los que los objetos tienen cientos de atributos. El clustering presenta grandes dificultades en estos espacios de grandes dimensiones, ya que el número de atributos irrelevantes crece de la misma forma que la dimensión, siendo necesario tomar alguna medida especial para controlar el agrupamiento. Dentro de esta clase de algoritmos se pueden enunciar los que siguen a continuación.

Clustering Subespacial (*Subspace Clustering*)

En esta sección se listan los algoritmos que intentan evitar la gran dimensionalidad, construyendo clusters en subespacios apropiados del espacio de atributos original.

El algoritmo CLIQUE (Clustering In QUEst) (Agrawal et al. 1998) se aplica para atributos numéricos, y es fundamental en el clustering subespacial.

El algoritmo ENCLUS (ENtropy-based CLUStering) (Cheng et al. 1999) sigue los mismos pasos que CLIQUE, pero usa un criterio diferente para la selección del subespacio. MAFIA (Merging of Adaptive Finite Intervals) (Goil et al. 1999), modifica significativamente CLIQUE.
El algoritmo OPTIGRID (Hinneburg & Keim 1999) realiza el particionado de datos mediante la división recursiva de celdas multi-dimensionales. Además usa las estimaciones de densidad de la misma forma que el algoritmo DENCLUE.
PROCLUS (PROjected CLUstering) (Aggarwal et al. 1999) asocia un subconjunto C con un subespacio de baja-dimensión, tal que la proyección de C en el subespacio es un cluster ajustado.
ORCLUS (ORiented projected CLUSter generation) (Aggarwal & Yu 2000) usa una aproximación, en cuanto a la proyección de clustering, similar al anterior pero no emplea subespacios paralelos de altas dimensiones.

Técnicas de Co-Clustering

Estas técnicas dividen los atributos en grupos similares, surgiendo nuevos atributos derivados que representan cada grupo. Se va realizando de forma simultanea el clustering de los puntos y el de los atributos. Para mejorar el clustering de los puntos (basándose en sus atributos), intenta clusterizar los atributos (basándose en los puntos). La idea del co-clustering de puntos de datos y atributos es antigua (Hartigan 1975) y se conoce bajo diferentes denominaciones: *simultaneous clustring, bi-dimensional clustering, block clustering, conjugate clustering, distributional clustering,* e *information bottleneck method.*

Para profundizar más sobre el tema se pueden consultar los trabajos de (Govaert 1995) y (Dhillon & Kogan 2002), así como los algoritmos Ping-Pong (Oyanagi et al. 2001) y SIMPLIFYRELATION (Berkhin & Becher 2002).

2.5. Resumen

Después de este recorrido por las diferentes clasificaciones de los modelos de estimación del coste del software, así como de las descripciones de los más importantes, se

tiene una visión bastante completa del estado actual de desarrollo de la materia. Esta visión ha permitido comprobar que dentro de los métodos de estimación paramétricos, hay cabida para una técnica que hasta ahora no se ha utilizado demasiado en el área de estimación del coste del software. Se trata de los métodos de clustering, los cuales serán empleados en este trabajo. Por tanto, para familiarizarse con ellos, se ha hecho una descripción de los principales algoritmos de este tipo, mostrando los más importantes.

El análisis del estado de la cuestión llevado a cabo permitirá, en el siguiente capítulo, estudiar la situación que ha llevado a la utilización de técnicas de clustering, en combinación con modelos de estimación paramétricos, para la estimación del coste de proyectos de software.

Capítulo 3

Planteamiento del problema

Lo que caracteriza a una inteligencia formada es que puede descansar satisfecha con el grado de precisión que la naturaleza de un asunto permite, y no buscar la exactitud cuando sólo una aproximación de la verdad es posible [...]

Aristóteles

En este capítulo se describen los problemas detectados en el proceso de estimación del coste en el desarrollo de un proyecto de software, derivados de la utilización, para el proceso de estimación, de datos de proyectos heterogéneos procedentes de una base de datos históricos. Todo el planteamiento está basado en la utilización de modelos paramétricos de estimación del coste del software. Estos modelos son los que serán utilizados en esta tesis, ya que los trabajos que han contribuido a su realización, se han basado en ellos.

3.1. Introducción

Para llevar a cabo un proyecto de software, es necesario realizar previo a su construcción, un conjunto de actividades encuadradas dentro de la *gestión de proyectos de software*. Antes de que el proyecto comience, el gestor y el equipo de software deben efectuar una estimación del trabajo a realizar, de los recursos necesarios y del tiempo que transcurrirá desde el comienzo hasta el final de su implementación. Siempre que se estima, se mira hacia el futuro y se acepta cierto grado de incertidumbre.

Aunque la estimación se podría considerar más un arte que una ciencia, es una actividad importante que debe llevarse a cabo con el máximo rigor, pues está en juego la propia viabilidad de la empresa.

Como se ha visto, las técnicas de estimación paramétricas tratan de buscar la relación subyacente existente entre diferentes atributos de un proyecto, considerados como cost drivers, y el tiempo de desarrollo y/o coste del mismo. Esta relación puede venir dada de muy diferentes maneras: como una función lineal, exponencial, potencia, etc. Para establecer esta relación se utilizan datos procedentes de proyectos realizados con anterioridad, de manera que el comportamiento de los mismos es el que influirá en las predicciones futuras. En ello radica la importancia de utilizar proyectos homogéneos, con unas características similares. Y claro está, el modelo de predicción obtenido solamente será válido cuando los proyectos, de los que se desea estimar el esfuerzo, tengan similares características a las de los proyectos utilizados en la obtención del predictor.

Es en este punto cuando se hace necesario realizar una reflexión acerca de la naturaleza de los proyectos, en cuanto a su cantidad y homogeneidad:

- Si se estudia, por ejemplo, el modelo COCOMO II, usado para la estimación del esfuerzo de desarrollo de software, se comprueba que el número de proyectos que utiliza es del orden de cientos, siendo la gran mayoría de ellos de características similares. Es decir, los proyectos no son muy numerosos, y además suelen ser homogéneos entre sí (Boehm, Horowitz, Madachy, Reifer, Clark, Steece, Brown, Chulani & Abts 2000).

- Por otra parte, en la actualidad, se suele disponer de bases de datos de proyectos con miles de ellos, cuya procedencia puede ser muy dispar. Por ejemplo, la base de datos de proyectos ISBSG versión 8 contiene 2028 proyectos, procediendo la información de éstos de muy diferentes fuentes (empresas de todo el mundo que pueden realizan actividades muy distintas entre sí).

Hasta ahora, las bases de datos utilizadas para la estimación del coste del software,

estaban constituidas por muy pocos proyectos, del orden de decenas a cientos de ellos. Además, estos proyectos solían ser homogéneos entre sí, debido a su procedencia (empresas del mismo país, o sector, o incluso de una única empresa). Por ello, el modelizar estas bases de datos mediante una única ecuación paramétrica, proporcionaba unos resultados satisfactorios la mayoría de las veces.

Es en la actualidad, con la aparición de organizaciones internacionales, tales como ISBSG, que recopilan información acerca de proyectos de software de empresas de todo el mundo, que pueden ser muy diferentes entre sí, cuando el modelizar todos estos proyectos mediante una única ecuación paramétrica proporciona unos resultados no todo lo satisfactorios que debería. El motivo reside principalmente en la heterogeneidad de los proyectos utilizados en la modelización.

Si se trata de obtener un modelo paramétrico, utilizando datos correspondientes a proyectos muy diferentes entre sí, por ejemplo, unos utilizan herramientas CASE mientras que otros no, o han sido realizados por grupos de desarrolladores constituidos por programadores con diferentes niveles de experiencia, o incluso la información disponible de los proyectos puede corresponderse tanto a proyectos de gestión como a proyectos en tiempo real, etc., es muy posible que el modelo trate de generalizar todo lo que pueda y se ajuste a la diversidad de proyectos, pero que no consiga representar de forma significativa a ninguno en concreto. Esto produciría una pobre estimación ante nuevos proyectos, porque como dice el refrán “aprendiz de mucho, maestro de nada”.

A continuación se profundizará en los problemas asociados a la estimación realizada utilizando proyectos de software heterogéneos. Posteriormente se destacará la influencia de los cost drivers sobre la estimación del coste realizada, terminando con un resumen del capítulo.

3.2. Heterogeneidad de los proyectos de software

El utilizar una ecuación de la forma $e = A_0 \cdot s^{A_1}$, donde e es el esfuerzo estimado, s alguna medida del tamaño del proyecto, y las constantes A_0 y A_1 se obtienen mediante análisis de regresión sobre una base de datos de proyectos históricos, donde esos proyectos proceden de diferentes fuentes, además de ser muy dispares entre sí a través de muchos de sus atributos, proporciona en ocasiones unos resultados pobres. Por ejemplo, utilizando la herramienta Reality de la base de datos de proyectos ISBSG 7, aplicada a 709 proyectos, se obtiene la siguiente ecuación:

$$e = 47,73 \cdot s^{0,76}$$

donde el esfuerzo se expresa en horas, y el tamaño en puntos de función. Un análisis de la bondad del ajuste ofrece como resultados MMRE=1,18 y PRED(0,3)=25,6 %. Ambas medidas son difícilmente aceptables, dado el alto grado de desviación sobre la inmensa mayoría de los datos.

La explicación dada para justificar estos resultados, sería que cuando se utilizan una serie de proyectos muy diferentes entre sí, la curva de regresión no captura las particularidades de cada uno de ellos, sino que trata de modelizar el conjunto total, perdiendo de esta forma precisión en la estimación.

Para ver de forma gráfica lo que aquí se describe, se puede ver la figura 3.1 en la que se utilizan 1546 proyectos[1] de la base de datos ISBSG versión 8. La curva de regresión obtenida (utilizando RCT) para estos proyectos aparece en la figura con la forma $e = 49,2 \cdot s^{0,68}$. Como se puede observar, esta curva representa a los puntos que hay junto a ella, que no son muchos en comparación con el total. La inmensa mayoría de los proyectos no se encuentran representados por esta curva ya que están muy alejados de la misma.

Si en vez de tener una base de datos de proyectos históricos heterogéneos, se tuviese una base de datos de proyectos homogéneos, los resultados serían mucho más precisos,

[1] En el capítulo 5 se verá porqué se utiliza esté número de proyectos.

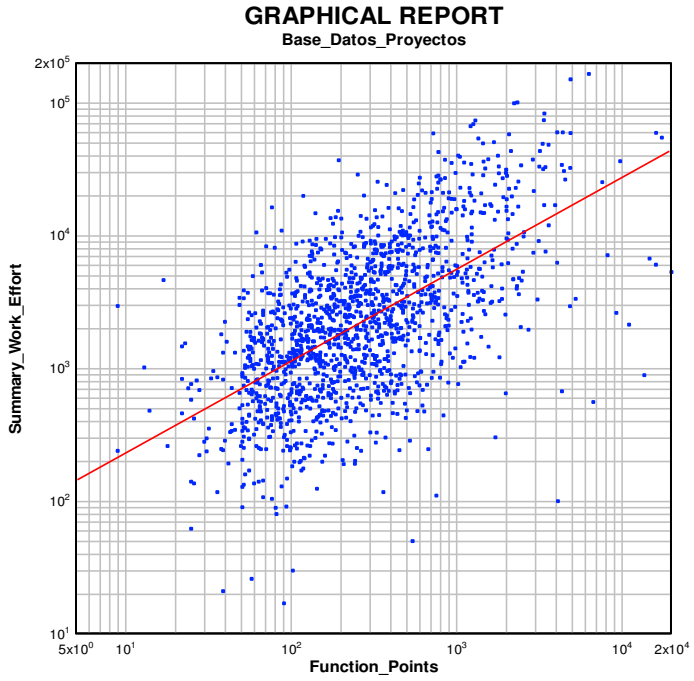

Figura 3.1: Curva de regresión que representa a todos los proyectos de la base de datos.

debido a que la curva de regresión obtenida capturaría de una forma más adecuada la naturaleza de tales proyectos (los cuales estarían acumulados a su alrededor), produciendo mejores estimaciones ante proyectos similares.

Es por ello por lo que surge la idea de intentar homogeneizar lo heterogéneo. Aparentemente dos términos contradictorios, sin embargo posibles de compatibilizar. Dentro de una base de datos de proyectos, existen grupos de ellos que tienen unas características internas similares, aunque diferentes a las de otros grupos de proyectos. Se trataría entonces de agrupar tales proyectos de manera que se obtuvieran diferentes grupos de proyectos. La heterogeneidad de los proyectos se manifestaría en los diferentes segmentos, mientras que los proyectos de cada uno de estos segmentos serían homogéneos entre sí.

Si se consiguiera agrupar proyectos de comportamiento parecido presentes en una

base de datos histórica, se obtendrían diferentes segmentos formados por un conjunto de proyectos homogéneos y que presentan diferentes propiedades a los proyectos que constituyen otros segmentos. De esta forma se tendrían segmentos formados por proyectos homogéneos, siendo estos segmentos disjuntos entre sí.

En esta nueva situación, que se consigue mediante la utilización de algoritmos de agrupamiento (clustering) conocidos (Garre et al. 2004), es posible obtener para cada uno de los segmentos así construidos, una ecuación paramétrica del tipo descrito anteriormente, obteniéndose una curva de regresión que describe, de forma más precisa, el comportamiento previsible de proyectos pertenecientes a cada uno de los grupos de proyectos.

A su vez, una vez establecidos los clusters presentes en la base de datos a un determinado nivel, se pueden volver a aplicar los algoritmos de agrupamiento dentro de cada uno de ellos, para poder determinar la presencia de nuevos clusters de un nivel inferior, procediendo así, de forma recursiva, mientras se aprecie una heterogeneidad significativa. Este proceso genera una jerarquía de clusters con un grado de homogeneidad interna elevado.

El objetivo de realizar este proceso recursivo de segmentación, se basa en la hipótesis de que las curvas de regresión que representan a los clusters así obtenidos, serán más representativas y se ajustarán mejor a los datos de cada segmento, pudiéndose obtener de esta manera mejores predicciones. La subdivisión de segmentos se puede realizar de forma recursiva hasta llegar a conseguir clusters indescomponibles, es decir, aquellos que no proporcionarían mejores resultados si se llevase a cabo su división.

Esta técnica será descrita en mayor detalle en el próximo capítulo.

En el trabajo de Erik Stensrud et. al (Stensrud et al. 2002) ya se comprueba la existencia de grupos de proyectos de diferentes comportamientos, definiendo el concepto

de *heteroscedasticity* (ya mencionado en el capítulo 2). A continuación, por considerarse de especial interés, se dedicará un subapartado al análisis de este trabajo.

3.2.1. Heteroscedasticity

Erik Stensrud et. al (Stensrud et al. 2002), en su trabajo *An Empirical Validation of the Relationship Between the Magnitude of Relative Error and Project Size* realizan un estudio acerca de la relación existente entre las medidas de evaluación de los modelos de estimación del coste del software (MRE, MMRE y PRED(l)), y el tamaño del mismo. Durante este análisis, suponen que los proyectos software presentan un patrón como el mostrado en la figura 3.2.

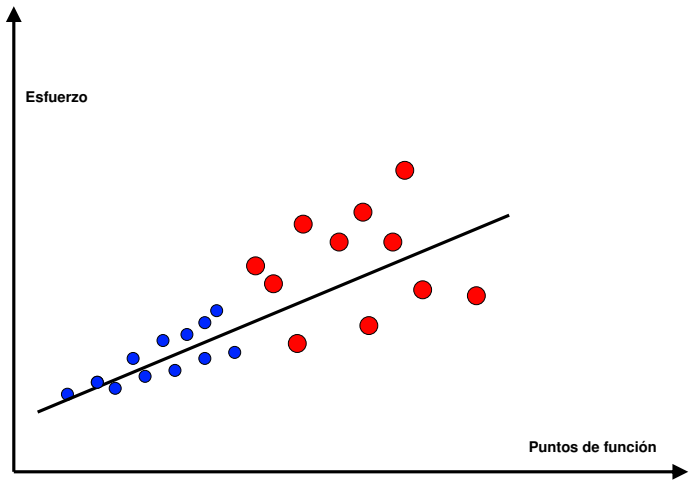

Figura 3.2: Conjunto de datos que presenta un patrón *heteroscedastic*.

Este patrón representa un conjunto de datos heteroscedastic, es decir, datos que tienen distinta varianza. En concreto, se observa como los proyectos más pequeños tienen una menor varianza que los proyectos grandes, los cuales se encuentran más alejados de la curva de regresión que los representa.

Este hecho lleva a los autores a realizar su estudio, dividiendo los proyectos en dos grupos, un grupo con proyectos de pequeño tamaño y otro grupo con proyectos de gran

tamaño. Para cada uno de ellos se obtiene su respectiva curva de regresión, y se lleva a cabo el análisis de la relación existente entre el tamaño de los proyectos y las medidas de evaluación de los modelos de estimación.

De esta forma se obtienen unas conclusiones más acertadas que si sólo se utilizase una única curva de regresión para todos los proyectos.

En este trabajo, los autores, ya observan un comportamiento diferente entre proyectos de distinto tamaño, llevándoles a la realización de estudios separados para ellos.

3.3. Influencia de los cost drivers sobre la estimación del coste del software

En modelos paramétricos, tales como el planteado por COCOMO II, la relación existente entre el esfuerzo y los diferentes cost drivers por los que se ve afectado, está ponderada por diferentes pesos. Estos pesos son mayores o menores, de acuerdo a la importancia que el modelo COCOMO II le da a los diferentes cost drivers que aparecen en esta relación. Uno de estos multiplicadores de esfuerzo es el *Uso de Herramientas Software (Use of Software Tools - TOOLS)*, que pertenece al modelo Post-Arquitectura de COCOMO II, y tiene únicamente en cuenta el grado de cobertura de la herramienta.

En el trabajo de Jongmoon Baik et. al (Baik et al. 1998) se extiende el concepto de TOOLS dentro de COCOMO II, en el sentido de tener en cuenta con respecto a las herramientas CASE, su grado de integración, madurez de la herramienta y soporte para el usuario. Los autores del trabajo demuestran que estos factores influyen significativamente en el esfuerzo de desarrollo del software.

La información ofrecida acerca de la influencia que ciertos cost drivers tienen sobre la productividad del software, en concreto la mejora que se obtiene al usar herramientas CASE, se puede tener en cuenta con el objeto de poder utilizarla para “orientar” el

proceso de agrupamiento de proyectos. Esta idea se verá descrita en detalle en el capítulo dedicado a la evaluación del modelo (capítulo 5).

3.4. Resumen

En este capítulo se han descrito los inconvenientes y carencias detectados en la utilización de una única ecuación de estimación para describir toda una base de datos de proyectos heterogéneos. Así mismo se han revisado un par de trabajos en los que se establece por un lado, la heteroscedasticity de los proyectos de software, y por otro, la influencia de los costs drivers sobre el esfuerzo en la construcción de software. Estos conceptos servirán de base para alguno de los desarrollos seguidos en esta tesis.

Las conclusiones que pueden extraerse de esta exposición permiten proponer soluciones a los problemas planteados, lo que se abordará en el siguiente capítulo de este trabajo.

Capítulo 4

Descripción del modelo paramétrico segmentado

Cualquier persona, por ignorante que sea, está convencida de poseer al menos una de las llaves del cofre de la sabiduría . . . y quizá esté en lo cierto.

Gil Bejes Sampao

En este capítulo se describe el modelo teórico de estimación, con el que se pretende mejorar el proceso de estimación paramétrica que se realiza utilizando bases de datos de proyectos heterogéneos. Se trata de un modelo que combina herramientas clásicas de inteligencia artificial con los modelos matemáticos tradicionales de estimación del coste del software.

4.1. Introducción

Ya se ha estudiado en el capítulo anterior la problemática existente en torno al proceso de estimación paramétrica de coste del software, al utilizar bases de datos que incluyen proyectos muy dispares entre sí. La solución propuesta pasa por la segmentación de la base de datos de proyectos y el tratamiento individual de cada uno de los grupos detectados.

En la sección 4.2 se realizará una descripción general de la técnica, destacando los principios teóricos en los que se basa. A continuación en la sección 4.3 se formalizará la descripción general realizada. Finalizando el capítulo con un pequeño resumen en la sección 4.4.

4.2. Descripción general del modelo

El modelo presentado en este trabajo se basa en la famosa frase de Julio Cesar "divide et vinces". En el caso que nos ocupa, el problema del que se parte sería la base de datos de proyectos heterogéneos, y los subproblemas que se obtienen tras un proceso de segmentación serían los grupos de proyectos homogéneos entre sí. El realizar el proceso de estimación sobre la base de datos original no produce resultados todo lo satisfactorios que se desearía, sin embargo si este proceso se realiza de forma individual para cada grupo de proyectos resultantes de la división, se mejora considerablemente el proceso de estimación (Garre et al. 2004, Garre et al. 2005).

Se trata por tanto de realizar el proceso de obtención de las ecuaciones de estimación en dos pasos:

1. Dividir la base de datos de proyectos en segmentos de proyectos de similares características entre sí.

2. Aplicar un modelo de estimación paramétrico para cada uno de los segmentos obtenidos, realizando los posteriores análisis sobre cada uno de los modelos obtenidos.

Tras este proceso, y en vista de los resultados obtenidos, es posible volver a segmentar de forma individual los clusters que se desee, con la intención de que los nuevos segmentos así obtenidos ofrezcan un mejor comportamiento. A cada uno de estos nuevos segmentos habría que aplicarle el modelo de estimación paramétrico correspondiente.

Igualmente, el proceso de estimación del esfuerzo de un nuevo proyecto estaría descompuesto en dos pasos:

1. Encontrar el cluster al que corresponde el nuevo proyecto a estimar.

2. Aplicarle las ecuaciones de estimación correspondientes a su cluster.

A continuación se describirán los pasos necesarios para obtener las ecuaciones de estimación. Sobre la forma de estimar el esfuerzo se profundizará en el capítulo 5.

4.2.1. Segmentación de proyectos

Para poder llevar a cabo la segmentación de proyectos, se utiliza una técnica de Inteligencia Artificial que realiza Aprendizaje Automático No Supervisado (*"Unsupervised Machine Learning"*), conocida como Clustering (Mitchell 1997).

Existen diferentes algoritmos de clustering, descritos en el capítulo 2, que realizan la agrupación de datos atendiendo a diferentes criterios de similitud entre los mismos. En el próximo capítulo se describirán en mayor detalle los algoritmos utilizados en este trabajo, así como los resultados obtenidos en su aplicación. Dependiendo del tipo de algoritmo, los grupos obtenidos pueden variar de forma significativa, y por lo tanto, los valores estimados que se derivan de su utilización.

A los algoritmos de clustering se les suministra un único conjunto de proyectos heterogéneos y tras realizar su trabajo proporcionan los diferentes grupos de proyectos, agrupados según el criterio de similitud utilizado por el algoritmo. Tal y como se ha comentado en el apartado anterior, es posible volver a segmentar los clusters así obtenidos mediante un algoritmo de clustering. Esta operación se puede volver a realizar de forma recursiva el número de veces que se desee hasta llegar a segmentos indescomponibles.

Estos segmentos indescomponibles, son aquellos segmentos que el correspondiente algoritmo de clustering no vuelve a dividir. El funcionamiento de estos algoritmos no permite que se realice una nueva descomposición, si la misma empeora la función objetivo que se está optimizando (maximizando o minimizando). De todo ello se puede concluir que desde el punto de vista técnico, nunca se llegará a obtener clusters de un único elemento.

Por otra parte, no hay que descartar la experiencia que se dispone acerca del mínimo número de proyectos "adecuado" para realizar un análisis de regresión. Se dispone de la información de los trabajos de diferentes autores, que utilizan bases de datos de proyectos con un número de ellos, por debajo del cual los resultados pasarían a ser poco

fiables. Por ejemplo, el conjunto de datos de *Albrecht* contiene 24 proyectos (Albrecht & Gaffney 1983), el conjunto de *Kemerer* contiene 15 proyectos (Kemerer 1987), y el de *Finnish* 40 proyectos (Kitchenham & Kansala 1993), proporcionando una idea del número de proyectos "adecuado" para su modelización paramétrica.

4.2.2. Modelización paramétrica de los segmentos obtenidos

Se trata ahora de realizar el proceso de modelización sobre cada uno de los segmentos de proyectos obtenidos en el paso anterior. Estos segmentos pueden ser de primer nivel, si no se ha realizado segmentación recursiva, o de cualquier otro nivel dentro de la jerarquía de segmentos, si es que se ha realizado un proceso recursivo de segmentación. De una u otra forma, se trata de segmentos que dentro de la jerarquía se encuentran en el último nivel dentro de su rama, es decir de *nodos* hoja o *terminales*.

Para construir el modelo paramétrico de cada uno de estos segmentos, se puede utilizar una función potencia de la forma $e = A_0 \cdot s^{A_1}$, donde e es el esfuerzo estimado, s una medida del tamaño del proyecto tal como puntos de función, y A_0 y A_1 son constantes obtenidas posteriormente mediante análisis de regresión. También se podría utilizar un modelo lineal que ajustase a una función de la forma $e = A_0 + A_1 \cdot s$, la cual se incluye como una posibilidad más en *Recursive Clustering Tool* (RCT).

Si tras aplicar el clustering, recursivo o no, se han obtenido k-segmentos, se tendrán entonces k-funciones paramétricas, donde los valores de las constantes ${A_0}^i$ y ${A_1}^i$ $(i = 1..k)$ se obtienen a partir de la curva de regresión de cada uno de los segmentos.

Cada una de estas k-funciones da lugar a un modelo de estimación, los cuales permiten realizar k-estimaciones distintas, una para cada uno de los segmentos de proyectos obtenidos. Para evaluar la calidad de cada una de estas estimaciones, se utilizan las medidas MMRE y PRED(l), las cuales serán descritas en mayor detalle en el próximo capítulo. Se tendrán entonces varias medidas de bondad, $MMRE_i$ y PRED(l)$_i$ (i=1..k), de cada uno de estos modelos.

4.3. Definición formal del modelo

Para describir el modelo de estimación propuesto en este trabajo, se procederá mediante una aproximación top-down. En primer lugar se hará una descripción general del mismo, independiente de la forma de la relación entre esfuerzo y los diferentes cost drivers, así como de las medidas de calidad del modelo. Una vez descrito el modelo general, se descenderá en el nivel de abstracción del mismo, detallando el modelo específico utilizado.

4.3.1. Modelo general

El modelo queda descrito según la siguiente relación:

$$e(p) = \bigwedge_{i=1}^{k} [f_i(p)] \tag{4.3.1}$$

Esta relación indica que el esfuerzo se puede expresar en función de cualquier atributo de un proyecto $e(p)$. Existen k-relaciones entre el esfuerzo y cualquier conjunto de cost drivers. Estas relaciones pueden tener diferentes formas y depender de diferentes atributos, de ahí que se exprese como $f_i(p)$. Por último, indicar que el operador $\bigwedge$, es un *"selector/agregador"*, y esto desde el punto de vista de la estimación del esfuerzo para un nuevo proyecto.

Selector: Se identifica el cluster al que se asignaría el proyecto a estimar, y se *selecciona* la función correspondiente a dicho cluster, que se utilizará en el cálculo del esfuerzo.

Agregador: Si se tiene en cuenta que un proyecto podría asignarse a varios segmentos, con diferente grado de pertenencia, ahora se podría obtener el esfuerzo del mismo como una *agregación* (combinación) de las funciones de cada uno estos segmentos.

Este operador se utiliza en esta tesis como *selector*, dejándose como trabajo futuro el poder contemplarlo en el otro sentido.

Para evaluar la calidad del modelo se puede utilizar como medida de calidad general:

$$q_i(B, e(p)) \tag{4.3.2}$$

Se trata de una medida de calidad para cada una de las k-relaciones ($i = 1 \dots k$) existentes en el modelo, y para su cálculo es necesario disponer de la base de datos de proyectos B, y de la correspondiente relación del esfuerzo $e(p)$ obtenida mediante la ecuación 4.3.1.

El algoritmo que proporciona el modelo general es el siguiente:

```
BUSCAR-MODELO(D, C, Q, m, A)
    ▷ D es la base de datos histórica
    ▷ C es el conjunto de cost drivers significativos
    ▷ Q son las restricciones de calidad deseadas para el resultado
    ▷ m es el modelo matemático a falta de ajustar
    ▷ A es el algoritmo que encuentra los segmentos s_i con i = 1 ... k
1   f ← ENCONTRAR-MODELO-LOCAL(D, C, m)
2   Si SATISFACE(D, Q, f) ∨ ¬SEGMENTABLE(D, C)
        Entonces
3           Devolver f
        Sino
4           ⟨s_1 ... s_k⟩ ← A(D, C)
5           Si k > 1
6           Para cada segmento s_i ∈ ⟨s_1 ... s_k⟩
             Hacer
7                   f_i ← BUSCAR-MODELO(s_i, C, Q, m, A)

8           Devolver UNIR-MODELOS(⟨f_1 ... f_k⟩)
```

Los procedimientos que aparecen en el algoritmo son:

ENCONTRAR-MODELO-LOCAL$(\mathcal{D}, \mathcal{C}, m)$**:** construye un modelo para $\mathcal{D}$ (corresponde al conjunto de proyectos que constituyen el parámetro real, con el que se realiza la llamada a esta función).

SATISFACE$(\mathcal{D}, \mathcal{Q}, f)$**:** esta función indica si el modelo obtenido satisface la medida de calidad deseada $\mathcal{Q}$.

SEGMENTABLE$(\mathcal{D}, \mathcal{C})$**:** devolverá falso si no es posible segmentar los proyectos representados por $\mathcal{D}$. Esta medida está estrechamente relacionada con el algoritmo de clustering a utilizar, o también es posible asociarla al criterio establecido por el conocimiento de que se disponga a partir de la experiencia.

UNIR-MODELOS$(\langle f_1 \dots f_k \rangle)$**:** se encarga de unir todos los modelos locales obtenidos durante el proceso recursivo de obtención de los mismos, en una única solución.

Además, este algoritmo necesita que se cumplan una serie de precondiciones:

1. El conjunto de datos $\mathcal{D}$ es una relación que contiene en su dominio, los elementos incluidos en el conjunto de cost drivers $\mathcal{C}$

2. El algoritmo de segmentación debe ser capaz de tratar con el tipo de cost drivers de $\mathcal{C}$, ya sean de tipo nominal, ordinal u otro

3. El modelo matemático m debe ser coherente con los cost drivers seleccionados

4. El algoritmo que encuentra los segmentos debe ser capaz de tratar con los diferentes tipos de cost drivers seleccionados. Por ejemplo, dado un algoritmo de clustering, éste debe poder manejar datos de tipo nominal, ordinal o real (según sea el cost driver), o realizar *ad hoc* algún tipo de procedimiento de discretización

4.3.2. Modelo específico

Se tienen k-segmentos o clusters, y para cada uno una función paramétrica de la forma descrita a continuación:

$Esfuerzo_1 = {A_0}^1 \cdot \text{Medida\_Tamaño}^{{A_1}^1}$
$Esfuerzo_2 = {A_0}^2 \cdot \text{Medida\_Tamaño}^{{A_1}^2}$
.....
$Esfuerzo_k = {A_0}^k \cdot \text{Medida\_Tamaño}^{{A_1}^k}$

El subíndice i-ésimo (i=1 ... k) indica el segmento representado por la función i-ésima, utilizándose como superíndice en las constantes correspondientes.

Como medidas de calidad para verificar la bondad del modelo se utilizan MMRE y PRED(l), en concreto para cada una de estas funciones se utilizan las siguientes medidas:

$MMRE_1$ y PRED(l)_1
$MMRE_2$ y PRED(l)_2
.....
$MMRE_k$ y PRED(l)_k

Los valores medios de MMRE y PRED(l), representativos del modelo paramétrico segmentado, se calculan como

$$MMRE = \frac{\sum_{i=1}^{k} MMRE_i}{k}$$

y

$$PRED(l) = \frac{\sum_{i=1}^{k} PRED(l)_i}{k}$$

Existen diferentes formas de utilizar estas medidas de calidad para evaluar el modelo paramétrico segmentado:

- Una manera podría consistir en la comparación de los valores de MMRE y PRED(l) obtenidos para una única función paramétrica, con los valores medios correspondientes al modelo paramétrico segmentado, mostrando así su calidad (Garre et al. 2004, Garre et al. 2005).

- Otra posibilidad consistiría en evaluar los modelos paramétrico y paramétrico segmentado, mediante la comparación de los valores de MMRE y PRED(l), obtenidos a partir de un proceso de estimación sobre un conjunto de proyectos de test.

Estas son las dos formas que se han utilizado en esta tesis para evaluar el modelo paramétrico segmentado. Respecto a la primera de ellas destacar que podría aplicarse alguna variante que tuviese en cuenta el tamaño de los clusters a la hora de realizar las medias y las comparaciones, esto se deja como trabajo futuro.

Un ejemplo gráfico del modelo se puede ver en la figura 4.1. Se ha optado por representar el tamaño del problema en puntos de función ya que la mayoría de los proyectos la base de datos ISBSG dispone de este dato, aunque se podría haber utilizado cualquier otra magnitud.

El gran círculo de la izquierda representa toda la base de datos de proyectos, la cual, tras pasar a través de algún algoritmo de clustering (representado por el rectángulo posicionado verticalmente), da lugar a una serie de segmentos de primer nivel. Si se desea, se puede volver a aplicar un algoritmo de clustering sobre alguno de los segmentos obtenidos, dando lugar finalmente en el caso del ejemplo a k-segmentos, estando cada uno de los cuales representado por un modelo de estimación paramétrico con su propia función.

El algoritmo en pseudocódigo, que permite obtener el modelo paramétrico segmentado, tal y como se ha descrito anteriormente, es el siguiente:

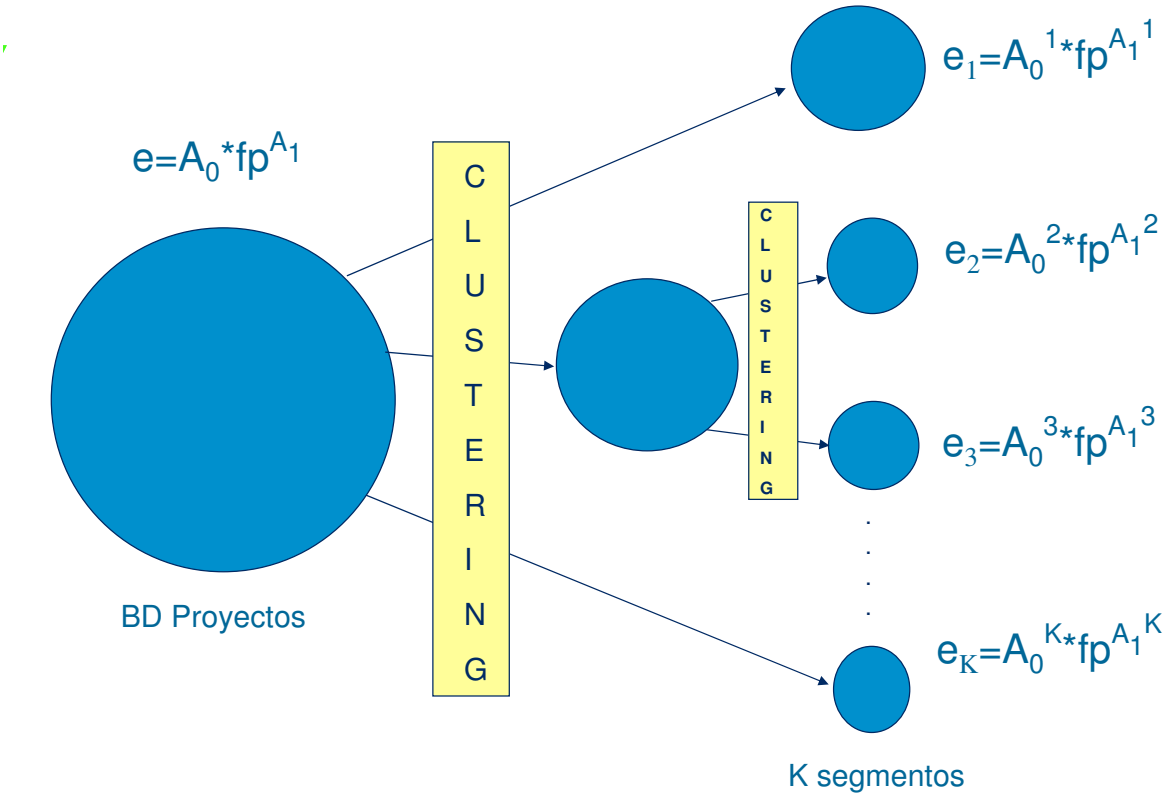

Figura 4.1: Descripción gráfica del modelo de estimación paramétrico segmentado

```
GENERAR-MODELO(𝒟, TIPO-MODELO, TIPO-CLUSTERING)
    ▷ 𝒟 es la base de datos histórica
    ▷ TIPO-MODELO es un modelo de estimación cualquiera
    ▷ TIPO-CLUSTERING es el algoritmo de clustering a utilizar
 1  SEGMENTOS:=OBTENER-SEGMENTOS(BD, TIPO-CLUSTERING);
 2  Para cada uno de los k-segmentos obtenidos
      Hacer
 3        ECUACION_i:=OBTENER-ECUACION(SEGMENTO_i, TIPO-MODELO);
 4        MMRE_i:=CALCULAR-MMRE(SEGMENTO_i, ECUACION_i);
 5        PRED(l)_i:=CALCULAR-PRED(SEGMENTO_i, ECUACION_i);
 6        Si NO se cumple CRITERIO(ECUACION_i, MMRE_i, PRED(l)_i)
            Entonces
 7              GENERAR-MODELO(SEGMENTO_i, TIPO-MODELO,
 8              TIPO-CLUSTERING);

 9        CONSTRUIR-SOLUCION(SEGMENTO_i, ECUACION_i, MODELO)

10  Devolver MODELO;
```

A continuación se describirán en detalle cada uno de los elementos que aparecen en el algoritmo:

BD: Son todos los proyectos agrupados en la base de datos de proyectos heterogéneos

TIPO-MODELO: Denota un modelo de estimación cualquiera. En este caso se trataría de un modelo paramétrico, tal y como se ha descrito en apartados anteriores

TIPO-CLUSTERING: Es el algoritmo de clustering que se utiliza. Como podrían utilizarse diferentes algoritmos, se indica de forma general

OBTENER-SEGMENTOS: Dada la base de datos de proyectos y el algoritmo de clustering a utilizar, obtiene la base de datos de proyectos segmentada

SEGMENTOS: Representa cada uno de los segmentos obtenidos tras realizar el proceso de clustering

SEGMENTO$_i$: representa el SEGMENTO i-ésimo, formado por los proyectos del cluster i-ésimo

OBTENER-ECUACION: Dados los proyectos de un cluster, y el tipo de modelo, ofrece la función, o forma correspondiente (según el tipo de modelo), que lo representa

ECUACION$_i$: Es la función i-ésima del modelo.

CALCULAR-MMRE: Obtiene el valor de MMRE para un segmento dado

MMRE$_i$: Valor MMRE para la función i-ésima

CALCULAR-PRED: Obtiene el valor de PRED(l) para un segmento dado

PRED(l):$_i$ Valor PRED(l) para la función i-ésima

CRITERIO: Es una función que determinará si es necesario dividir, a su vez, el segmento obtenido o no, en base a un determinado criterio. Este criterio puede ser automático, o establecido de forma manual. Básicamente se tendrían en cuanta variables como tamaño del segmento, valores mínimos de MMRE y PRED(l), así como otras cualquiera que se quieran tener en cuenta.

CONSTRUIR-SOLUCION: Va confeccionando poco a poco la solución conforme va obteniendo segmentos. En general la solución estará formada por todas las funciones de cada uno de los segmentos obtenidos, junto con sus parámetros, valores MMRE y PRED(l) así como datos relativos a las características de los segmentos. Todo ello formará la solución, dada por MODELO.

Para finalizar este capítulo, hay que destacar que tanto el algoritmo general como el específico, son algoritmos probabilistas, es decir, ante los mismos datos de entrada no

siempre producen los mismos resultados. Esto es así debido a la naturaleza de algunos algoritmos de clustering, los cuales no siempre inicializan sus parámetros de la misma forma, con lo que pueden llevar a segmentaciones diferentes en distintas ejecuciones.

4.4. Resumen

En este capítulo se ha descrito el modelo paramétrico segmentado, como solución propuesta a los problemas encontrados en el proceso de estimación. Así mismo se ha mostrado el algoritmo que permite llegar a obtener dicho modelo, tanto el general como el específico. El siguiente capítulo aborda la evaluación del modelo presentado, de cara a la posterior comprobación de la consecución de los objetivos planteados en el primer capítulo.

Capítulo 5

Evaluación

Son vanas y están plagadas de errores las ciencias que no han nacido del experimento, madre de toda certidumbre.

Leonardo Da Vinci

Una vez definido el modelo paramétrico segmentado, y presentada su forma de aplicación, se aborda la evaluación del mismo. El objetivo fundamental es permitir examinar la consecución de los objetivos planteados en el primer capítulo del presente trabajo. La evaluación que se presenta, se ha llevado a cabo a través de la realización de una serie de experimentos sobre la base de datos de proyectos ISBSG versión 8. Los resultados obtenidos se ofrecen a modo de evaluación del modelo construido.

5.1. Metodología general llevada a cabo para realizar la evaluación del modelo segmentado

Es necesario describir, de una forma clara y estructurada, la manera en la que se ha llevado a cabo la evaluación del modelo de estimación segmentado, con el fin de ayudar a entender el proceso completo seguido. Para ello se describirán las decisiones tomadas así como sus justificaciones, agrupadas en las siguientes secciones:

- Base de datos de proyectos

  Se verá el proceso de preparación de datos seguido para proporcionar una base de datos con datos lo más fidedignos posible.

- Análisis de regresión

  Es necesario hacer una pequeña introducción al fundamento matemático que se

necesita para establecer la relación existente entre el esfuerzo y los puntos de función.

- Estudio del caso base
  Como parte de la evaluación del modelo segmentado, se realizarán comparaciones con los resultados obtenidos a través del modelo que utiliza una única función paramétrica (algunas veces denotado por modelo único) para realizar la estimación del esfuerzo.

- Selección de los algoritmos de clustering utilizados
  En esta sección se describirán los algoritmos de clustering que se han utilizado, así como los motivos que han llevado a ello.

- Criterios para la evaluación del modelo paramétrico segmentado
  Se justifican y describen los criterios utilizados para evaluar la calidad que ofrece el modelo paramétrico de cara a la estimación del esfuerzo de proyectos de software.

- Ejecución y análisis de resultados
  Se mostrarán los resultados obtenidos tras la realización de una serie de pruebas, junto con la justificación de su realización.

5.2. Base de datos de proyectos

Para la realización de todo el trabajo que forma esta tesis, se ha utilizado la base de datos de proyectos de desarrollo software de la organización *International Software Benchmarking Standards Group* (ISBSB), ya que contiene gran cantidad de proyectos y permite, por lo tanto, la realización de las pruebas que aquí se han realizado, aprovechando la heterogeneidad de estos proyectos.

ISBSG es una organización sin ánimo de lucro, fundada en 1997 como consecuencia de la colaboración, durante varios años, de un grupo de asociaciones estadounidénses dedicadas al estudio de métricas del software. El objetivo de esta organización consiste en ayudar, a empresas tanto públicas como privadas, en la gestión de los recursos de

las tecnologías de la información, mediante el suministro y explotación de bases de datos públicas que almacenan conocimiento sobre ingeniería del software. Estas bases de datos están estandarizadas y verificadas, y son representativas de las tecnologías de la información más actuales.

Actualmente se dispone de una base de datos de proyectos de desarrollo y mejora de software (versión 9), que contiene unos 3000 proyectos. Esta información se ha conseguido recopilar gracias a la colaboración, durante años, de empresas de desarrollo de software de todo el mundo.

ISBSG proporciona productos y servicios que permiten a los desarrolladores de productos software y personal de mantenimiento:

- Estimar, de una forma más exacta, el esfuerzo, tiempo y coste.
- Disminuir el riesgo de desarrollo.
- Incrementar la productividad.
- Etc.

La versión de la base de datos de ISBSG utilizada en este trabajo es la 8, la cual contiene información sobre 2028 proyectos. Esta base de datos contiene información sobre tamaño, esfuerzo, y otras muchas características de cada proyecto. Para su utilización ha sido necesario realizar un proceso previo sobre la misma. Este proceso ha consistido en la realización de una limpieza, eliminando algunos proyectos. Además se ha realizado un análisis de consistencia sobre las etiquetas de los valores de los atributos, realizando algunas modificaciones sobre éstos, que aún siendo los mismos se llaman de diferente manera en distintos proyectos. La inconsistencia de la base de datos puede llevar a engañar a los algoritmos de clustering que tratarían elementos iguales como diferentes, dando lugar, incluso a la generación de un número mayor de clusters. A continuación se describe el proceso seguido:

Eliminación de proyectos: El primer paso de limpieza consistió en la eliminación de todos los proyectos cuya calidad es dudosa, esto es, los que tienen para el atributo "Data Quality Rating" un valor igual a "C" o "D", sólo se tienen en cuenta los proyectos con valores para este atributo igual a "A" o "B", ya que éstos indican fiabilidad de los datos. A continuación se eliminaron todos los proyectos con valor para el atributo "Recording Method" distinto de *Staff Hours*. Este atributo indica la forma en que se ha medido el esfuerzo, siendo este valor el más fiable, el resto de formas de considerar el esfuerzo son subjetivas. Por ejemplo *Productive Time* es una magnitud difícil de valorar en un contexto organizativo. Otro aspecto a tener en cuenta para la limpieza de los datos es la forma en la que se obtuvieron los diferentes valores de los puntos de función. En concreto se examinó el valor del atributo "Derived count approach", descartando todos los proyectos que no hubiesen utilizado como forma de estimar los puntos de función métodos como IFPUG, NESMA, Albretch o Dreger. Las diferencias entre los métodos IFPUG y NESMA tienen un impacto despreciable sobre los resultados de los valores de los puntos de función (*NESMA FPA. Counting Practices Manual (CPM 2.0)* 1996). Las mediciones basadas en las técnicas Albretch no se eliminaron ya que, de hecho IFPUG es una revisión de estas técnicas. De la misma forma el método Dreger (Dreger 1989) es simplemente una guía sobre las mediciones IFPUG.

Consistencia de valores: Es necesario hacer una revisión de los valores de los atributos para unificar denominaciones. Por ejemplo, hay campos vacíos, mientras que en otros pone "don't know", se dejan todos en blanco. Tales campos vacíos no revisten problema alguno, se tratan como tales. Hay otros casos en los que, por ejemplo se denomina al lenguaje COBOL 2 o COBOL II, es necesario nombrarlos de la misma forma, ya que se trata del mismo lenguaje, en este caso COBOL 2. No se van a describir aquí todas las modificaciones realizadas, quédese únicamente con el proceso realizado.

Para obtener información sobre los atributos que forman la base de datos de proyectos,

se puede consultar la página de la organización ISBSG[1].

Los atributos que se han utilizado para la evaluación del modelo han sido:

Function Points: Representa el valor de los puntos de función, ajustados por el *Value Adjustment Factor*.

Summary Work Effort: Proporciona el esfuerzo total en horas, necesario para llevar a cabo el proyecto de software.

Upper CASE Used: El valor de este atributo indica si el proyecto utiliza *upper CASE tools*, herramientas CASE que soportan las actividades de desarrollo del software posteriores a la implementación.

Used Methodology: Establece si se utilizó alguna metodología en la construcción del proyecto.

Estos atributos se nombran de manera informal como: *fp*, *e*, *CASET* y *METHO*, respectivamente. Los atributos *fp* y *e* se utilizan en el proceso de clustering, es decir, para segmentar los proyectos. Son atributos numéricos, ideales para los algoritmos de clustering aquí utilizados, y aparecen con valores no nulos para todos los proyectos de la base de datos ISBSG. En cuanto a *CASET* y *METHO*, se utilizarán, en un proceso de particionado previo de los proyectos, antes de ser segmentados.

Una vez realizado todo el proceso de limpieza y consistencia de los datos, la base de datos queda con un total de 1546 proyectos, los cuales serán los utilizados en las pruebas de la sección 5.7.

El formato de los archivos utilizados para realizar el proceso de segmentación ha sido el *ARFF*[2] (*Attribute Relation File Format*), propio del grupo de desarrollo de Weka (Ian & Eibe 2000). La razón de su utilización estriba en darle una mayor portabilidad y

[1] http://www.isbsg.org/.
[2] http://www.cs.waikato.ac.nz/∼ml/weka/arff.html.

compatibilidad, sobre todo con Weka.

Un fichero ARFF es un fichero de texto ASCII que describe una lista de instancias que comparten un conjunto de atributos. Estos ficheros están divididos en dos secciones, la primera es la **cabecera**, seguida de la información sobre los **datos**.

La **cabecera** de un fichero ARFF contiene el nombre de la relación que describe, una lista de los atributos (las columnas en las que se dividen los datos), y sus tipos. Por ejemplo, parte de la sección cabecera del archivo utilizado en este trabajo sería:

```
@RELATION Base_Datos_Proyectos

@ATTRIBUTE Derived_Count_Approach {IFPUG, NESMA, Albrecht}
@ATTRIBUTE Function_Points Numeric
@ATTRIBUTE Summary_Work_Effort Numeric
@ATTRIBUTE Resource_Level Numeric
@ATTRIBUTE Max_Team_Size Numeric
...
```

El primer atributo *Derived_Count_Approach*, es de tipo nominal, indicándose entre paréntesis los posibles valores que puede tomar. Los demás atributos son numéricos.

Los **datos** del fichero ARFF aparecerían como a continuación:

```
@DATA

IFPUG,74,1100,1,?,Enhancement,?,3GL,COBOL2,?,?,?,?,Yes,?,Billing;,?,?
IFPUG,832,20975,1,?,Enhancement,MF,3GL,C,?,?,?,?,Yes,?,Ordering;,?,?
IFPUG,183,789,1,?,Enhancement,?,?,Coldfusion,?,?,?,?,Yes,?,Sales&Marketing,?,?
IFPUG,135,7290,1,?,Enhancement,?,3GL,C,?,?,?,?,Yes,?,Voice_Provisioning,?,?
...
```

donde el símbolo "?" indica que el valor es nulo, y las comas separan los diferentes valores de los atributos. Cada línea se corresponde con una instancia (un proyecto en este caso).

5.3. Análisis de regresión

El **análisis de regresión** es la parte de la estadística que investiga la relación entre dos o más variables relacionadas entre sí. En el caso que ocupa este trabajo se trata de buscar la relación existente entre esfuerzo y puntos de función, según la relación potencia:

$$e = A_0 \cdot fp^{A_1} \tag{5.3.1}$$

o la función lineal:

$$e = A_0 + A_1 \cdot fp \tag{5.3.2}$$

donde e es la variable dependiente que representa el esfuerzo, A_0 y A_1 parámetros a determinar, y fp la variable independiente que representa los puntos de función.

Para poder realizar el estudio de regresión de fórmulas de tipo potencia se suele linealizar el modelo, de manera que los cálculos se puedan realizar de una forma más sencilla. Por ejemplo, esta función potencia se podría linealizar mediante las siguientes transformaciones:

$$e' = log(e),\ fp' = log(fp) \tag{5.3.3}$$

obteniéndose la forma lineal:

$$e' = log(A_0) + A_1 \cdot fp' \tag{5.3.4}$$

y a partir de ésta realizar todos los cálculos. Se trataría de una trasformación logarítmica doble de las dos variables implicadas. Lo que en *Recursive Clustering Tool* (RCT) se ha denominado *Linear log-log Plot*.

En otras ocasiones se aborda el problema de la solución de la fórmula 5.3.1 de forma directa, mediante algoritmos numéricos más complejos, concretamente se usa el método de Levenberg-Marquardt (Marquardt 1963). En RCT se ha denominado esta forma de

cálculo *Non-Linear Power*.

Para realizar una regresión de forma lineal como la fórmula 5.3.2, se hace según los métodos tradicionales de cálculo, sin demasiada complejidad. Simplemente con la palabra *Linear* se ha denotado este método de cálculo en RCT.

El modelo de regresión obtenido a partir de un conjunto de pares $(e_1, fp_1), \ldots (e_n, fp_n)$, consistirá en una curva de regresión, alrededor de la cual estarán distribuidos los puntos observados. Cuanto más cerca estén estos puntos de la curva obtenida, mejor será el modelo dado. Este compromiso de mejora es el que motiva el *principio de mínimos cuadrados* (*least squares*). Según este principio, una curva proporciona un buen ajuste a los datos si las distancias verticales (desviaciones) desde los puntos observados a la curva son pequeñas. La medida de la bondad del ajuste es la suma de los cuadrados de estas desviaciones. Entonces, la curva de mejor ajuste es la que tiene la suma de desviaciones al cuadrado más pequeña posible.

El método de mínimos cuadrados es el más extendido en su utilización, sin embargo cuando las gráficas u otras evidencias sugieren que el conjunto de datos contienen valores extremos o atípicos, es necesario tratarlos de un modo diferente, existiendo dos posibilidades:

1. Una posibilidad consistiría simplemente en omitir dichos puntos. Esto sería correcto si se encuentra que los puntos extremos resultaron de errores al registrar valores de datos o errores experimentales. Si esto no es así, no se podría asegurar con total certeza que su omisión fuese correcta.

2. Otro método consistiría en mantener los posibles puntos extremos, pero usar el método de estimación que asigna relativamente menos peso en valores distantes que el principio de mínimos cuadrados. Uno de estos principios es el de *mínimas desviaciones absolutas* (*least absolute deviation*). Este principio requiere un método iterativo de cálculo, encuadrado dentro de los denominados "métodos robustos de

estimación".

La herramienta RCT, descrita en el anexo A, trata de contemplar todas estas situaciones, posibilitando la utilización de cinco aproximaciones diferentes para obtener las respectivas curvas de regresión de los segmentos obtenidos. Éstas se describen a continuación:

Linear (least-absolute-deviation). Obtiene el ajuste de los puntos mediante la fórmula 5.3.2, utilizando, para minimizar el error cometido, el método de la mínima desviación absoluta. Es adecuada cuando existe una fuerte dependencia lineal entre los atributos dependientes e independientes de la fórmula y su distribución en el espacio aparece con muchos puntos dispersos entre sí.

Linear (least-squares). El ajuste de los puntos se realiza mediante la fórmula 5.3.2, utilizando ,para minimizar el error cometido, el método de mínimos cuadrados. Esta aproximación es adecuada cuando existe una fuerte dependencia lineal entre los atributos dependientes e independientes de la fórmula, no existiendo demasiada dispersión entre los puntos que ajusta.

Linear log-log Plot (least-absolute-deviation). Utiliza la fórmula 5.3.1 para ajustar los proyectos de software en su forma linealizada mediante doble transformación logarítmica, ecuaciones 5.3.3 y 5.3.4. Así mismo minimiza la desviación de los puntos hacia la curva mediante el criterio de mínima desviación absoluta. Esta aproximación es adecuada cuando la dependencia entre las variables dependientes e independientes no es lineal, y aparecen muchos puntos dispersos en el espacio.

Linear log-log Plot (least-squares). Es igual a la anterior, pero utiliza como método de minimización del error cometido el método de mínimos cuadrados.

Non-Linear Power (least-squares). Intenta ajustar los puntos según la fórmula 5.3.1, sin realizar ningún tipo de transformación sobre ella. Así mismo utiliza mínimos cuadrados. Esta forma daría resultados parecidos a la anterior.

Esta disponibilidad permitirá realizar multitud de combinaciones diferentes, con el objetivo de conseguir siempre el mejor ajuste de los proyectos observados.

5.4. Estudio del caso base

El caso base viene representado por el modelo paramétrico que, para la estimación del esfuerzo en la construcción de un proyecto de software, utiliza una única fórmula de la forma indicada en 5.3.1 o 5.3.2. Si se utiliza la base de datos de proyectos ISBSG versión 8, preparada tal y como se describe en la sección 5.2, sobre 1246 proyectos de entrenamiento y 300 proyectos de test, teniendo en cuenta las variables esfuerzo y puntos de función, es posible ajustar los 1246 proyectos mediante las diferentes aproximaciones de regresión proporcionadas por RCT. En la tabla 5.1 se muestran los diferentes valores de MMRE y PRED(0,3) obtenidos al utilizar diferentes regresiones. Destacar que para que los análisis comparativos sean consistentes, se han realizado las diferentes regresiones sobre el mismo conjunto aleatorio de 1246 proyectos de entrenamiento.

			Predicción de la Curva		Predicción del Modelo	
	A_0	A_1	MMRE	PRED(0,3)	MMRE	PRED(0,3)
Linear (least-absolute-deviation)	1041,36	3,6	1,68	23,86 %	1,36	22 %
Linear (least-squares)	3578,98	3,4611	4,11	14,76 %	3,48	13,34 %
Linear log-log Plot (least-absolute-deviation)	38,94	0,7343	1,58	22,44 %	1,41	22,34 %
Linear log-log Plot (least-squares)	41	0,7174	1,52	22,97 %	1,33	23 %
Non-Linear Power (least-squares)	207,15	0,5653	3,43	18,05 %	3,09	16 %

Tabla 5.1: Resultados de la aplicación de todos los modelos de regresión sobre 1246 proyectos de entrenamiento, y utilización de 300 de test

Estos resultados se han obtenido mediante la herramienta RCT, construida por el equipo de investigación de este proyecto. Los valores de A_0 y A_1 se corresponden con los parámetros de las ecuaciones 5.3.1 y 5.3.2, según se trate de regresión exponencial o lineal respectivamente. Para obtener estos valores se utiliza *v-fold cross validation*[3] (Zhang 1993), explicada en la sección 5.4.1, resultando además unos valores de MMRE y PRED(0,3) que denotan la capacidad predictiva de la curva obtenida. Por otra parte, el modelo se puede probar con los proyectos de test, obteniéndose en ese caso otros valores diferentes de MMRE y PRED(0,3), correspondientes a la capacidad predictiva del modelo. Ambas formas de comparar serán descritas en la sección 5.6.4.

Como se puede apreciar en la tabla 5.1 los valores de MMRE y PRED(0,3) en ambos

[3]Se utilizará el término inglés.

casos, predicción de la curva y predicción del modelo, son bastante similares. Ello puede ser debido a que en el proceso de obtención de los parámetros de la curva de regresión, aplicando v-fold cross validation, se van realizando sucesivos test con proyectos del mismo conjunto de entrenamiento, constituyendo los 300 proyectos de test con los que posteriormente se evalúa la capacidad predictiva del modelo una muestra representativa de las mismos.

Es de destacar que estos resultados son susceptibles de sufrir variaciones, ya que los 1246 proyectos de entrenamiento y los 300 de test, se han elegido de forma aleatoria y para una nueva ejecución podrían ser diferentes. Así mismo es también posible variar estos valores si se elige un número diferente de *splits*[4] para realizar v-fold cross validation.

5.4.1. Precisión de los modelos obtenidos

Para evaluar un modelo producido es posible utilizar *cross-validation*[5], para ello se seleccionan dos conjuntos, uno de entrenamiento y otro de test. Con el de entrenamiento se obtiene el modelo y con el de test se determina su precisión. Este método tiene un inconveniente y es que los datos de test no se pueden utilizar para obtener el modelo, y en el caso de que el número total de elementos no sea demasiado elevado, el modelo obtenido no será todo lo bueno que se desearía, ya que no sería totalmente representativo de la muestra existente, sólo de parte.

Para evitar este inconveniente se utiliza v-fold cross validation, la cual consiste en:

1. Se divide el conjunto de datos en v subconjuntos (splits) de tamaño similar, $v > 10$

2. Se establece uno de esos subconjuntos como conjunto de test

3. Se construye el modelo con los restantes conjuntos

[4]Este término significa divisiones o partes, aunque no será traducido.
[5]Se utilizará indistintamente el término inglés o su traducción al castellano.

4. Se valida el modelo obtenido con el conjunto de elementos de test, obteniéndose una medida de su precisión

5. Se establece el siguiente subconjunto como conjunto de test y se vuelve al paso 2

6. Una vez que se han utilizado todos los subconjuntos como conjuntos de test, se calcula la precisión media, siendo este valor la precisión del modelo obtenido utilizando todos los datos disponibles

La gran ventaja de v-fold cross validation frente a cross-validation es que se utilizan todos los datos disponibles para obtener el modelo, sin necesidad de reservar ningún conjunto de los mismos para evaluación. Esto es muy importante en áreas tales como Ingeniería del Software, en las que con mucha frecuencia los datos disponibles no son muy numerosos.

5.5. Selección de los algoritmos de clustering utilizados

En este estudio se han utilizado diversos algoritmos de clustering, en concreto COBWEB, k-medias y EM. COBWEB pertenece a la familia de algoritmos de clustering jerárquicos, mientras que los otros dos son algoritmos de la misma familia (métodos de particionado), aunque con algunas variaciones tal y como se describe en el trabajo de Berk (Berkhin 2002).

De las distintas familias de algoritmos, las más adecuadas para el tipo de datos a manejar son aquellas a las que pertenecen los algoritmos anteriormente mencionados. El resto no han presentado unas características aplicables a los atributos de proyectos. Por ejemplo, los modelos basados en datos categóricos no han sido considerados, los cuales no son característicos de las estimaciones paramétricas. Los basados en restricciones tampoco serían aplicables a los datos bajo estudio, ni los algoritmos para datos de grandes dimensiones (no es el caso), y así para el resto de familias.
Se comenzó utilizando el algoritmo COBWEB. Su comportamiento no fue demasiado

satisfactorio, ya que proporcionó como resultado un único cluster. Por ello sus resultados no se mostrarán, ya que serían muy parecidos a los ofrecidos en la tabla 5.1, ni se volverá a incidir sobre él.

Respecto a los algoritmos de particionado, k-medias es uno de los más conocidos y utilizados (Hartigan 1975, Hartigan & Wong 1979), mostrando un buen comportamiento para atributos numéricos. Por otro lado, los algoritmos probabilísticos, como EM son aplicables tanto a atributos numéricos como categóricos, caracterizándose por su flexibilidad y capacidad de configuración con diversidad de combinaciones a través del cambio de los valores de sus parámetros (Mitchell 1997). Se empezó utilizando EM, ya que es fácilmente accesible a través de la herramienta Weka[6], aunque posteriormente y por limitación de los resultados ofrecidos por la misma, se comenzó la construcción de una herramienta propia que contemplase el algoritmo EM.

Existen otras herramientas que se podrían haber probado, tales como AUTOCLASS, pero se ha preferido dejar de lado en una primera aproximación.

A continuación se procederá a la descripción de los algoritmos utilizados.

5.5.1. K-medias

Se trata de un algoritmo clasificado como Método de Particionado y Recolocación. El método de las k-medias (Hartigan 1975, Hartigan & Wong 1979), es hasta ahora el más utilizado en aplicaciones científicas e industriales. El nombre le viene porque representa cada uno de los clusters por la media (o media ponderada) de sus puntos, es decir, por su centroide. Este método únicamente se puede aplicar a atributos numéricos, y los outliers le pueden afectar muy negativamente. Sin embargo, la representación mediante centroides tiene la ventaja de que tiene un significado gráfico y estadístico inmediato. La suma de las discrepancias entre un punto y su centroide, expresado a través de la distancia apropiada, se usa como función objetivo.

[6]http://www.cs.waikato.ac.nz/∼ml/weka/.

La función objetivo, suma de los cuadrados de los errores entre los puntos y sus centroides respectivos, es igual a la varianza total dentro del propio cluster. La suma de los cuadrados de los errores se puede racionalizar, como el negativo del log-likelihood, para modelos mixtos que utilicen distribuciones normales. Por lo tanto, el método de las k-medias se puede derivar a partir del marco probabilístico (ver subsección *Clustering Probabilístico* del libro de Mitchell (Mitchell 1997)).

Existen dos versiones del método de las k-medias. La primera es parecida al algoritmo EM, y se basa en dos pasos iterativos: primero reasigna todos los puntos a sus centroides más cercanos, y en segundo lugar recalcula los centroides de los nuevos grupos creados en el anterior. El proceso continua hasta alcanzar un criterio de parada (por ejemplo que no se realicen nuevas reasignaciones). Esta versión se conoce como algoritmo de Forgy (Forgy 1965).
La segunda versión (Duda & Hart 1973) reasigna los puntos basándose en un análisis más detallado de los efectos causados sobre la función objetivo al mover un punto de su cluster a otro nuevo. Si el traslado es positivo, se realiza, en caso contrario se queda como está.

A diferencia de los algoritmos, COBWEB y EM, k-medias necesita la previa especificación del número de clusters que se desea obtener.

5.5.2. EM

Se mostrará en primer lugar qué son los modelos FM.

Finite Mixture Models (FM)

Estos modelos se usan en problemas de estimación de Funciones de Densidad de Probabilidad (FDP) de forma semi-paramétrica, lo cual se puede extrapolar a tareas de clustering. La FDP desconocida a la que pertenecen el conjunto completo de datos, se puede aproximar mediante una combinación lineal de NC componentes, definidas a

falta de una serie de parámetros $\{\Theta\} = \cup\{\Theta_j \forall j = 1..NC\}$, que son los que hay que averiguar,

$$P(x) = \sum_{j=1}^{NC} \pi_j p(x; \Theta_j) \text{ con } \sum_{j=1}^{NC} \pi_j = 1 \tag{5.5.1}$$

donde π_j son las probabilidades *a priori* de cada cluster, cuya suma debe ser 1, que también forman parte de la solución buscada, $P(x)$ denota la FDP arbitraria y $p(x; \Theta_j)$ la función de densidad del componente j. Cada cluster se corresponde con las respectivas muestras de datos que pertenecen a cada una de las densidades que se mezclan.

Se pueden estimar FDP de formas arbitrarias, utilizándose FDP de muy diversa naturaleza tales como normales n-dimensionales, t-Student, Bernoulli, Poisson, y log-normales. En este trabajo se contempla la modelización de los datos mediante distribuciones normales, distribuciones log-normales y t-Student.

El ajuste de los parámetros del modelo requiere alguna medida de su bondad, es decir, cómo de bien encajan los datos sobre la distribución que los representa. Este valor de bondad se conoce como el likelihood de los datos, ya explicado en el capítulo 2. Se trataría entonces de estimar los parámetros buscados maximizando este likelihood (este criterio se conoce como *ML-Maximun Likelihood* (ML)). Normalmente, lo que se calcula es el logaritmo de este likelihood, conocido como log-likelihood ya que es más fácil de calcular de forma analítica. La solución obtenida es la misma, gracias a la propiedad de monotonicidad del logaritmo. La forma de esta función log-likelihood (Dempster et al. 1977) es:

$$L(\theta, \pi) = \log \prod_{n=1}^{NI} P(x_n) \tag{5.5.2}$$

donde NI es el número de instancias, que se suponen independientes entre si.

En este proceso existen dos aspectos a resolver:

Estimación del valor de los parámetros: Como se ha comentado anteriormente, para obtener θ se debe maximizar L. Este máximo se podría obtener de forma analítica, dependiendo de la forma de $p(x;\theta_j)$. Se calcularían las derivadas de L, se igualarían a cero y se resolvería el sistema de ecuaciones, que finalmente proporcionaría los valores de θ. Lo que sucede es que en la práctica, esta aproximación es intratable. Por ello se suele utilizar el método iterativo *Expectation-Maximization (EM)* (Hartly 1958, Dempster et al. 1977, McLachlan & Krishnan 1997), el cual da una aproximación de *ML*, permitiendo estimar los parámetros característicos de cada distribución que compone la función de distribución mixta.

Estimación del número de componentes: Para estimar el número de componentes no se puede utilizar el criterio *ML*, ya que cuantos más componentes se tengan mejor valor tendrá *ML* (*ML* es una función no decreciente de NC). Existen varias aproximaciones alternativas:

- Utilizar el algoritmo EM. Se utiliza el algoritmo EM para diferentes valores de NC, obteniendo para cada uno de ellos unas estimaciones de parámetros. El mejor valor de NC se define como el que minimice una cierta función de coste.
- Utilizar una función de coste que incluya la función L junto un término adicional que penalice valores grandes de NC. Una decisión óptima en este caso sería utilizar el criterio *Minimum Description Length* (MDL) (Rissanen 1983). Esto ayudaría a evitar el conocido overfitting[7] (sobreaprendizaje, sobreajuste).
- Otros esquemas se basan en realizar remuestreo, así como utilizar cross-validation (Ian & Eibe 2000, Mitchell 1997).

Decisiones de diseño

En la implementación del algoritmo EM se han tomado determinadas decisiones respecto a ciertos aspectos.

[7]Se utilizará el término en inglés.

Es posible seleccionar la forma de obtener NC, ya sea mediante v-fold cross validation, o mediante *MDL*. V-fold cross validation permite elegir el número de partes en las que se quieren dividir los datos (datos de entrenamiento y datos de test), mientras que en *MDL* se toman todos los datos para realizar el proceso. Los resultados ofrecidos por ambas posibilidades son muy parecidos, siendo mucho más rápido *MDL*. Para estimar los parámetros buscados θ, se aplica el algoritmo EM utilizando todo el conjunto de datos y el valor de NC obtenido anteriormente. Hay que recordar que cuantos más parámetros se tengan mayor overfitting aparecerá, por lo que se tenderá a un equilibrio.

Respecto a $p(x;\theta)$, se sabe que su elección admite varias alternativas, la herramienta implementada permite utilizar distribuciones normales N-dimensionales, (N se corresponde con el número de atributos):

$$f(x\,;\mu,\Sigma) = \frac{1}{|\Sigma|^{1/2}\ (2\pi)^{N/2}}\ \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right] \tag{5.5.3}$$

distribuciones log-normales:

$$f(x) = \frac{1}{x\sigma\prime\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{ln(x)-\mu\prime}{\sigma\prime}\right)^2} \tag{5.5.4}$$

donde

$$\mu\prime = ln(\mu) - \frac{1}{2}ln\left(\frac{\sigma^2}{\mu^2}+1\right) \tag{5.5.5}$$

$$\sigma\prime = \sqrt{ln\left(\frac{\sigma^2}{\mu^2}+1\right)} \tag{5.5.6}$$

donde μ y σ son la media y desviación típica, respectivamente de la función de distribución.

Y por último la distribución t-Student. Si $(X, X_1, X_2, \ldots, X_n)$ son $n+1$ variables aleatorias normales independientes de media 0 y varianza σ^2, la variable definida como

$$Y_n = \frac{X}{\sqrt{\frac{1}{n}\sum_{i=1}^{n} X_i^2}} \tag{5.5.7}$$

se dice que tiene una distribución t_n-Student con n grados de libertad. Su función de densidad es

$$f(x) = \frac{\Gamma\left(\frac{n+N}{2}\right)}{(n\pi)^{\frac{N}{2}}|\Sigma|^{\frac{1}{2}}\Gamma\left(\frac{n}{2}\right)}\left[1+\frac{1}{n}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]^{-\frac{n+N}{2}} ,\forall x \in \mathbb{R}, \tag{5.5.8}$$

siendo

$$\Gamma(p) = \int_0^\infty z^{p-1}e^{-z}dz \tag{5.5.9}$$

la función gamma de Euler, con $p > 0$.

En general se puede decir que esta distribución, para un número no muy grande de grados de libertad, sirve para representar a poblaciones de datos que tienen valores dispersos, o alejados de la media.

Proceso de clustering

El proceso mediante el cual, se obtiene una serie grupos de proyectos a partir de una base de datos inicial, requiere la ejecución de dos pasos:

1. En un primer lugar es necesario establecer el número de clusters, NC, en los que se agruparán los proyectos. Para ello se puede recurrir a dos aproximaciones diferentes:

   a) Utilizar v-fold cross validation.

   b) Utilizar el criterio MDL

2. Una vez que se conoce el número de clusters en los que segmentar la base de datos, es necesario obtener los parámetros del *Finite Mixture Model*.

Dada la naturaleza e importancia de este proceso, se estudiará aparte en la siguiente sección.

5.5.3. Proceso de clusterización llevado a cabo por la herramienta RCT

Como se ha comentado en el apartado anterior, es necesario realizar dos pasos, uno para obtener el número de clusters y otro para obtener los parámetros correspondientes. A continuación se verá cada uno de ellos.

Obtención del número de clusters NC

Para obtener el número de clusters en los que se segmentará la base de datos de proyectos es posible hacerlo de dos formas distintas, mediante validación cruzada o utilizando el criterio MDL.

Validación Cruzada. Este proceso consiste en realizar v-fold cross validation, seleccionando diferentes conjuntos de test y de entrenamiento, llevando a cabo de forma iterativa los pasos *expectation* y *maximization* hasta que se cumpla un cierto criterio de parada. Estos pasos se describen a continuación.

Paso 1: Como tratamiento previo de los datos, se efectúa una reordenación aleatoria de todas las instancias. Se comienza fijando *NC=1*.

Paso 2: Se dividen los datos en v partes de igual tamaño.

Paso 2.1: Se toma una de las partes como conjunto de *Test*. El resto, las otras *v-1* partes, forman el conjunto *Training*.

Paso 2.2: Training

- Obtención de los parámetros iniciales θ_k^0, π_k^0 para $k = 1, \ldots, NC$: se suma el valor de cada uno de los atributos de las NI_{Train} instancias del conjunto *Training*. Se ordenan las instancias de acuerdo a este valor y a continuación se divide el conjunto así obtenido en NC partes iguales. Para cada una de estas partes se calcula

$$\mu_k^0 = \frac{\sum\limits_{n \in k} x_n}{NI_{Train}/NC} \ ,$$

$$\Sigma_k^0 = \frac{\sum\limits_{n \in k} (x_n - \mu_k^0)(x_n - \mu_k^0)^T}{NI_{Train}/NC} \ ,$$

$$\pi_k^0 = \frac{NC}{NI_{Train}} \tag{5.5.10}$$

- Iteración *ML-EM*: Se aplica el algoritmo *EM*, estimando en cada iteración i la función log-likelihood $L(\theta^i, \pi^i)$, que crecerá hasta alcanzar un valor máximo *ML*. El criterio usado para detener la iteración, descrito por varios autores (Ian & Eibe 2000), es que en diez iteraciones consecutivas la variación de este valor no sea mayor de 10^{-10}. Brevemente, los pasos de la iteración EM son:

  - Paso E:

$$w_{kn}^i, = \Pr(\{\theta_k^i\ ,\ \pi_k^i\}/x_n) = \frac{\pi_k^i\, f\left(x_n\,;\,\mu_k^i, \Sigma_k^i\right)}{\sum_{l=1}^{NC} \pi_l^i\, f\left(x_n; \mu_l^i, \Sigma_l^i\right)} \tag{5.5.11}$$

    donde $k = 1, \ldots, NC$ and $n = 1, \ldots, NI_{Train}$

  - Paso M:

$$\mu_k^{i+1} = \frac{\sum_{n=1}^{NI_{Train}} w_{kn}^i x_n}{\sum_{n=1}^{NI_{Train}} w_{kn}^i}\ ,$$

$$\Sigma_k^{i+1} = \frac{\sum_{n=1}^{NI_{Train}} w_{kn}^i (x_n - \mu_k^{i+1})\,(x_n - \mu_k^{i+1})^T}{\sum_{n=1}^{NI_{Train}} w_{kn}^i}\ ,$$

$$\pi_k^{i+1} = \frac{\sum_{n=1}^{NI_{Train}} w_{kn}^i}{NI_{Train}} \tag{5.5.12}$$

Paso 2.3 Test

Los resultados del paso anterior $\{\theta_k,\ \pi_k\ \textit{para k=1,...,NC}\}$, se aplican al conjunto formado por las NI_{Test} instancias de *Test*, usando como indicador de validez, o medida del error cometido al predecir dichas instancias, el propio log-likelihood, cambiado de signo, pues es una medida del error

$$E_s = -\sum_{n=1}^{NI_{Test}} \log \sum_{k=1}^{NC} \pi_k\, f\left(x_n\,;\,\mu_k, \Sigma_k\right) \tag{5.5.13}$$

siendo $s = 1 \ldots v$.

Paso 2.4 Volver al paso 2.1 y repetir los pasos 2.2 y 2.3 con otros conjuntos hasta completar las v combinaciones que se pueden realizar (obsérvese que los conjuntos de *Test* son disjuntos).

Paso 2.5 Una vez realizado todo el proceso de v-fold cross validation para un NC determinado, se calcula un valor promedio de los errores cometidos

$$\overline{E} = \frac{\sum_{s=1}^{v} E_s}{v} \tag{5.5.14}$$

el cuál se utilizará más adelante para obtener el número de clusters óptimo.

Paso 3: Volver al paso 2 con $NC = 2, 3, \ldots$ hasta un número máximo de clusters elegido de forma manual NC_{max}, de modo que permita monitorizar el mínimo buscado, pero que cumpla la siguiente condición que limita el número total de parámetros $M < NI \cdot N/2$, calculándose M como

$$M = NC\left(1 + N + \frac{N\,(N+1)}{2}\right) - 1 \tag{5.5.15}$$

así, el número estimado de clusters, y los correspondientes parámetros asociados, vendrá dado por

$$NC_{opt} = argmin_{NC}\left\{\overline{E}\right\} \tag{5.5.16}$$

donde $NC = 1, ..., NC_{max}$.

Criterio MDL. En este caso no se utilizan conjuntos separados de datos de entrenamiento y de test, sino que todos los datos son utilizados para el proceso.

Paso 1: Como tratamiento previo de los datos, se efectúa una reordenación aleatoria de todas las instancias. Se comienza fijando *NC=1.*

Paso 2: EM

- Obtención de los parámetros iniciales θ_k^0, π_k^0 para $k = 1, \ldots, NC$: se suma el valor de cada uno de los atributos de las NI instancias de todo el conjunto de

proyectos. Se ordenan las instancias de acuerdo a este valor y a continuación se divide el conjunto así obtenido en NC partes iguales. Para cada una de estas partes se calcula

$$\mu_k^0 = \frac{\sum\limits_{n \in k} x_n}{NI/NC} \ ,$$

$$\Sigma_k^0 = \frac{\sum\limits_{n \in k} (x_n - \mu_k^0)(x_n - \mu_k^0)^T}{NI/NC} \ ,$$

$$\pi_k^0 = \frac{NC}{NI}$$

- Iteración *ML-EM*: Se aplica el algoritmo *EM*, estimando en cada iteración i la función log-likelihood $L(\theta^i, \pi^i)$, que crecerá hasta alcanzar un valor máximo *ML*. El criterio usado para detener la iteración, descrito por varios autores (Ian & Eibe 2000), es que en diez iteraciones consecutivas la variación de este valor no sea mayor de 10^{-10}. Los pasos de la iteración EM fueron ya descritos anteriormente, sólo que esta vez los sumatorios se refieren a todas las instancias del conjunto de proyectos, no sólo a los datos de entrenamiento.

Paso 3: Para el presente número de clusters, se construye el indicador *MDL* de la forma siguiente

$$MDL(NC) = -\sum_{n=1}^{NI} \log \sum_{k=1}^{NC} \pi_k f\left(x_n; \mu_k, \Sigma_k\right) + \frac{1}{2} M \log\left(NI \cdot N\right) \qquad (5.5.17)$$

donde NI son todas las instancias y M es el número total de parámetros definido en la ecuación 5.5.15.

Paso 4: Volver al paso 2 con $NC = 2, 3, \ldots$ hasta un número máximo de clusters elegido de forma manual NC_{max}, de modo que permita monitorizar el mínimo buscado, pero que cumpla la siguiente condición que limita el número total de parámetros $M < NI \cdot N/2$. Así, el número estimado de clusters, y los correspondientes parámetros asociados, vendrá dado por

$$NC_{opt} = argmin_{NC} \left\{MDL(NC)\right\} \qquad (5.5.18)$$

donde $NC = 1, ..., NC_{max}$.

Obtención de los parámetros

Dependiendo de la procedencia de NC, los parámetros se obtendrán, a su vez, de dos formas distintas.

Validación cruzada. Es necesario volver a aplicar el algoritmo EM, según los pasos descritos para el apartado correspondiente a la obtención de NC, pero teniendo en cuenta que ahora los sumatorios se extienden a la totalidad de las instancias NI. El paso 2.2 ofrece el valor de los parámetros buscados, una vez llegados al número de clusters que coincide con el óptimo NC_{opt}.

Criterio MDL. En este caso no hay que realizar ningún proceso adicional, el mismo procedimiento seguido para obtener NC lleva a la obtención del valor de los parámetros. Cuando se haya llegado al número de clusters óptimo NC_{opt}, el paso 2 ofrecerá los valores de dichos parámetros.

5.6. Criterios para la evaluación del modelo paramétrico segmentado

En esta sección se describen las medidas de calidad que se usarán para evaluar todos los aspectos del modelo paramétrico segmentado.

Respecto al proceso seguido para obtener el modelo, existen dos aspectos a evaluar:

1. Por una parte es necesario evaluar la calidad de las curvas de regresión dadas para cada uno de los clusters obtenidos

2. Y por otra parte hay que evaluar el modelo completo, formado por todos los segmentos

A continuación se mostrará la forma en la que se realizarán las comparaciones entre el modelo paramétrico y el modelo paramétrico segmentado. Para terminar con un apartado dedicado a la forma de predecir el esfuerzo para futuros proyectos que se deseen construir.

5.6.1. Evaluación del proceso de clustering

Como medidas de calidad se utilizan MMRE (*Mean Magnitude of Relative Error*) y PRED(l) (*Prediction level*). Conte et al. (Conte et al. 1986) considera como aceptable para un modelo de estimación un valor de $MMRE \leq 0,25$. PRED(l) indica la proporción de observaciones para las que MRE es menor o igual a l. Conte et al., considera un valor de $PRED(0,25) \geq 0,75$, como un criterio aceptable de la calidad de un modelo de estimación. Se pueden usar diferentes valores para l, en este trabajo se utiliza $l = 0,30$, aunque puede tomar otros valores tales como $l = 0,25$. MRE se define para una observación como:

$$MRE_i = \frac{|ValorActual_i - ValorPredicho_i|}{ValoActual_i} \tag{5.6.1}$$

la forma de MRE para N observaciones se caracteriza por la *Mean MRE* (MMRE):

$$MMRE = \frac{1}{N}\sum_{i=1}^{N}\frac{|ValorActual_i - ValorPredicho_i|}{ValoActual_i} \tag{5.6.2}$$

El nivel de predicción se define como:

$$PRED(l) = \frac{k}{N} \tag{5.6.3}$$

siendo k el número de observaciones para las que MRE es menor o igual a l.

Como medida de la calidad de un modelo de estimación se puede usar una u otra, sin embargo en este trabajo se ha querido utilizar ambas, para dar una mayor información.

Hay dos aspectos diferentes que es necesario evaluar por separado, por una parte hay que comprobar la calidad de cada curva de regresión obtenida, y por otra parte hay que evaluar el modelo de estimación.

5.6.2. Evaluación de las curvas de regresión

Existen dos posibilidades a la hora de evaluar las curvas de regresión obtenidas, por un lado se puede comprobar la *capacidad predictiva* de cada una de ellas, mientras que por otra parte se puede comprobar cómo de bien se ajustan a éstas los datos utilizados para su obtención. Veamos a continuación cada una de ellas.

Encaje de las curvas de regresión

Por encaje se entiende la adecuación de las curvas a los puntos que se han utilizado para obtenerlas. Para comprobar lo bien que las curvas representan a esos datos es necesario utilizar, tanto para su entrenamiento como para su posterior testeo, el mismo conjunto de puntos. Esto es posible realizarlo a través de la opción *Curve Fitting*, tal y como se describirá en el anexo A, de la herramienta RCT.

Puede ser interesante tener esta medida en cuenta, para saber con qué tipo de curvas, más o menos precisas, se realizará el posterior análisis predictivo, y de esta forma tener más información para poder sacar conclusiones.

Capacidad predictiva de las curvas de regresión

La forma de comprobar la calidad predictiva de las curvas de regresión obtenidas se realiza mediante v-fold cross validation, siendo igualmente posible, a través de la opción *Predictive Accuracy (Number of splits)* de la herramienta RCT, seleccionar el número de particiones en las que se dividirán los datos de entrenamiento y de test. Por ejemplo un valor de 3 para el número de splits indicaría que se utilizan el $33,\hat{3}\,\%$ de instancias para test y el $66,\hat{6}\,\%$ para entrenamiento, realizándose 3 veces validación cruzada simple con 3 conjuntos diferentes de datos de test y de entrenamiento.

El proceso de segmentación, que como se ha comentado en diferentes secciones puede ser recursivo, proporciona diferentes curvas de regresión, una para cada uno de los segmentos *terminales*[8] obtenidos. Estas curvas se evalúan con los datos de test correspondientes, obteniendose los diferentes valores de MMRE y PRED(0,3) asociados a cada una de ellas. A partir de estos valores, es posible calcular los correspondientes valores medios de MMRE y PRED(0,3), que representan una medida de todas ellas en conjunto.

Los valores individuales de MMRE y PRED(0,3) de cada uno de los segmentos, ofrecen información acerca del comportamiento predictivo local de cada uno de ellos,

[8]Finales en la jerarquía arbórea de clusters generada.

mientras que los valores medios de MMRE y PRED(0,3) dan información acerca del comportamiento predictivo medio de todos los modelos paramétricos que representan a los diferentes segmentos.

5.6.3. Evaluación de las estimaciones llevadas a cabo por el modelo sobre valores de test

Para realizar el ensayo del modelo completo, es necesario disponer de un método para determinar el cluster al que pertenece cada instancia a probar, de forma que se pueda recorrer el árbol completo y llegar hasta el nodo final donde se aplicará la correspondiente curva de regresión para estimar el parámetro deseado. En el caso de haber particionado[9] el cluster en función del valor de cierto atributo la asignación es trivial. En el caso en el que se haya utilizado el algoritmo EM para segmentar, la cuestión es más complicada. A continuación se describe el procedimiento empleado.

El criterio de pertenencia, empleado en este trabajo, es el de la mínima distancia de la instancia a probar, respecto de las medias de cada cluster en el espacio $\{A_1, A_2, \ldots A_p\}$, donde los A_i son un subconjunto de dimensión p de los atributos, pero no usando la métrica euclídea, sino la de Mahalanobis. Nótese que este subconjunto de atributos no tiene por qué coincidir con los atributos utilizados en el proceso de clustering.

Del proceso de clustering se obtienen las probabilidades de cada instancia de entrenamiento condicionadas a la pertenencia a cada componente, esto es $p(x_i/C_j)$, donde x_i es cada una de las instancias de entrenamiento, $i = 1, 2, \ldots, N_{\text{Train}}$, y C_j es cada uno de los componentes obtenidos del proceso de clustering, $j = 1, 2, \ldots, NC$, con NC número de clusters obtenidos. Con esta información, una estimación de las medias, varianzas y covarianzas de cada componente en el espacio anteriormente definido serían

[9]Dividir en tantas partes como posibles valores pueda tomar un atributo categórico.

$$\mu_j^l = \frac{\sum_{i=1}^{N_{\text{Train}}} p(x_i/C_l) \cdot x_i^j}{\sum_{i=1}^{N_{\text{Train}}} p(x_i/C_l)} \tag{5.6.4}$$

$$\Sigma_{j,k}^l = \frac{\sum_{i=1}^{N_{\text{Train}}} p(x_i/C_l) \cdot (x_i^j - \mu_j^l) \cdot (x_i^k - \mu_k^l)}{\sum_{i=1}^{N_{\text{Train}}} p(x_i/C_l)} \tag{5.6.5}$$

para $j,\ k = 1, 2, \ldots, p$ y $l = 1, 2, \ldots, NC$.

Una instancia de prueba y_i pertenece al cluster C_s con $1 \leq s \leq NC$, si se cumple

$$min\{(y_i - \mu^l)[\Sigma^l]^{-1}(y_i - \mu^l),\quad l = 1, 2, \ldots, NC\} \text{ para } l = s \tag{5.6.6}$$

La expresión anterior se puede plantear en definitiva, como una serie de inecuaciones que definen regiones de pertenencia en el espacio $\{A_1, A_2, \ldots, A_p\}$. En el supuesto de que uno de los atributos del citado espacio sea el que se pretende conocer (normalmente el *esfuerzo*), se debe hacer una estimación inicial de este, utilizando la curva de regresión del conjunto de datos, antes de ser segmentados, tal y como se indica en la descripción del funcionamiento de RCT.

A modo de ejemplo, las regiones del espacio *esfuerzo, puntos de función* que se definen en el nodo "cluster 6" del caso práctico de la sección 5.7.3, son las que se muestran en la figura 5.1.

Según puede apreciarse en la figura 5.1, la instancia de prueba se asignará al cluster correspondiente a la región en la que esté situada.

Como caso particular, cuando se usa un espacio unidimensional $\{A\}$, se tendría como criterio una serie de intervalos en A construidos de forma sencilla a partir de las medias y las varianzas de cada cluster respecto a este atributo.

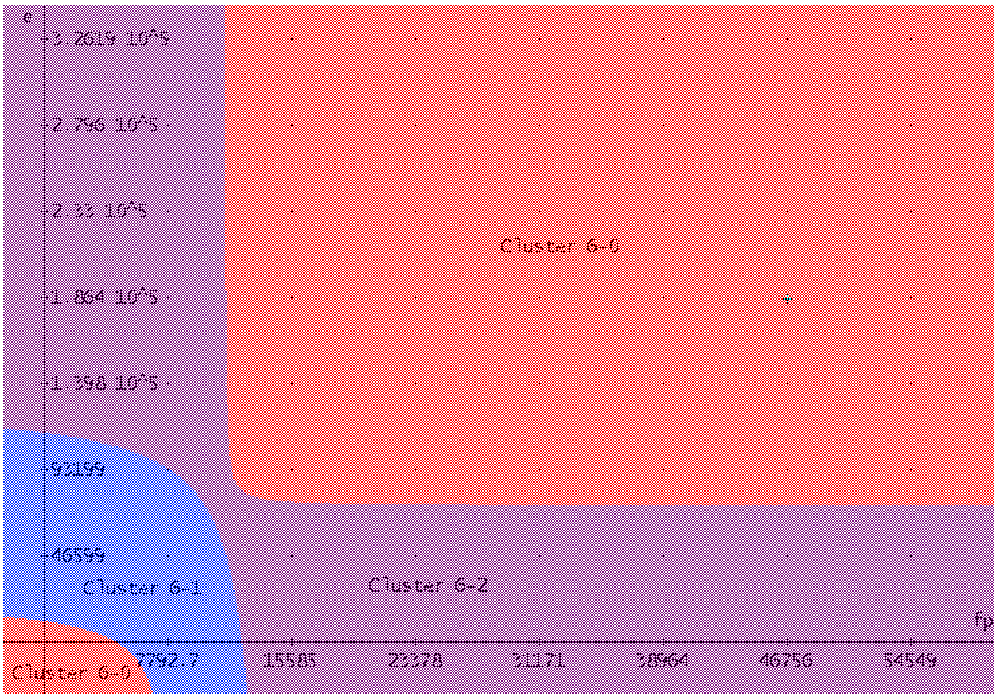

Figura 5.1: Ejemplo de regiones en el plano *e,fp*

Por ejemplo, suponiendo que A fuesen puntos de función y se tuviesen únicamente dos clusters, aparecerían dos intervalos definidos de la siguiente forma, $0 \leq fp \leq P$ y $P < fp \leq \infty$, donde P se define como:

$$P = \mu_1 + \frac{1}{1 + \frac{\sigma_2}{\sigma_1}}(\mu_2 - \mu_1) \tag{5.6.7}$$

siendo μ_i y σ_i, $i = 1, 2$, las medias y desviaciones típicas de cada uno de los dos clusters. Para saber la ecuación que se debe utilizar para estimar el esfuerzo (desconocido) de la instancia de prueba, se haría de la siguiente manera: para un valor de puntos de función comprendido en el intervalo $[0, P]$ $(0 \leq fp \leq P)$ se utilizaría la curva del primer cluster, mientras que para valores pertenecientes al intervalo (P, ∞) $(P < fp \leq \infty)$ se seleccionaría la curva del segundo cluster, siendo $P \in \Re$. Y todo esto según el criterio de aplicación de la ecuación 5.6.6.

La interpretación gráfica de esta asignación se puede apreciar en la figura 5.2. El

valor de D se corresponde con:

$$D = \frac{1}{1 + \frac{\sigma_2}{\sigma_1}}(\mu_2 - \mu_1) \tag{5.6.8}$$

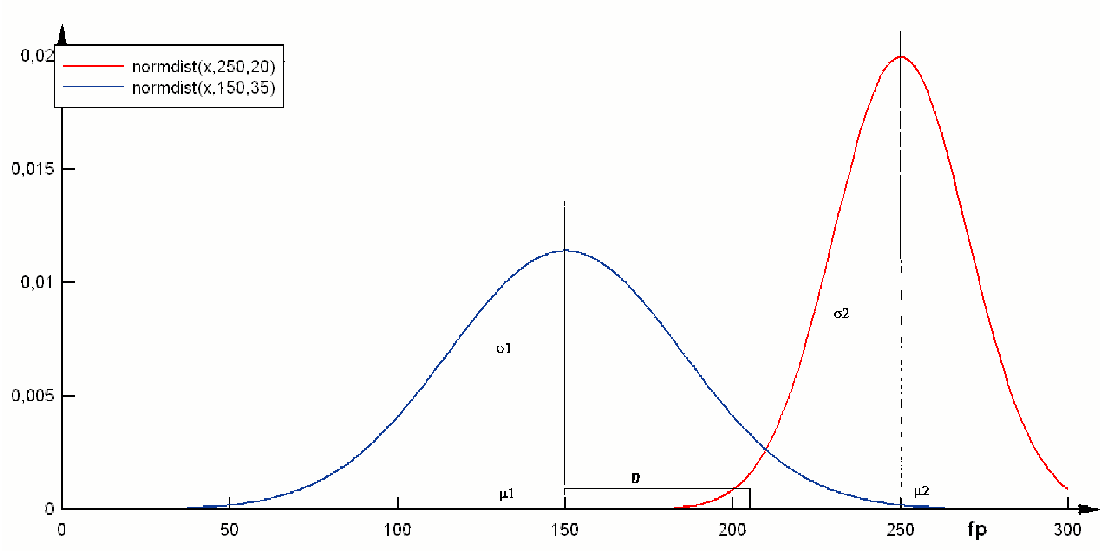

Figura 5.2: Funciones de distribución de dos de los clusters obtenidos

Sin pérdida de generalidad se puede extrapolar este ejemplo para NC clusters, formándose los intervalos ordenando los valores de las medias μ_i correspondientes a cada cluster, $i = 1, 2, \ldots, NC$ de menor a mayor. Las asignaciones se harían de la misma forma indicada anteriormente para el caso particular de 2 clusters.

Al aplicar este criterio para cada una de las instancias de test, se obtienen los correspondientes valores estimados del esfuerzo, utilizándose éstos para calcular los valores de MMRE y PRED(0,3). Estas medidas permiten considerar la capacidad predictiva del modelo paramétrico segmentado obtenido.

5.6.4. Comparación de las evaluaciones realizadas

Uno de los objetivos de este trabajo consiste en comparar los modelos de estimación que utilizan para estimar el esfuerzo una única ecuación paramétrica, que describe a toda la base de datos de proyectos, con una nueva aproximación que obtiene diferentes ecuaciones paramétricas para cada uno de los segmentos que se obtienen al clusterizar la base de datos de proyectos.

Cuando se calcula mediante v-fold cross-validation, la curva de regresión que define a todos los proyectos de la base de datos, se obtienen unos valores de MMRE y PRED(0,3), que de alguna manera indican la capacidad predictiva de dicha curva (sección 5.6.2) con respecto al esfuerzo. Es posible además obtener nuevos valores de MMRE y PRED(0,3), sección 5.6.3 sobre una serie de proyectos de test, de manera que se determine la capacidad predictiva de todo el modelo.

Cuando se segmenta la base de datos de proyectos y se calculan las diferentes curvas de regresión de cada uno de los segmentos resultantes, cada pareja de valores MMRE y PRED(0,3) de cada curva indica su *capacidad predictiva local*, siendo sus valores medios una medida de *capacidad predictiva media* de todas las curvas de regresión del modelo segmentado.

Igualmente, al testar el modelo segmentado como se indicó en la sección 5.6.3, los valores de MMRE y PRED(0,3) así obtenidos constituirán también una medida de *capacidad predictiva* pero de todo el modelo paramétrico segmentado.

En este trabajo se ofrecerán datos de todas estas capacidades, para cada uno de los experimentos realizados. Respecto al modelo segmentado, por un lado se detallarán los valores de MMRE y PRED(0,3) de cada uno de los segmentos obtenidos, por otra parte se mostrarán también sus valores medios, y finalmente los valores de MMRE y PRED(0,3) que se obtienen al probar el modelo con un conjunto de proyectos de test. Para el modelo único se mostrarán los valores de MMRE y PRED(0,3) de la curva de regresión obtenida, así como los mismos para los proyectos de test aplicados.

De esta forma se tienen dos medidas claramente comparables:

1. Por un lado se dispone de los valores medios de MMRE y PRED(0,3) de las curvas de regresión de los clusters obtenidos, los cuales ofrecen una medida de *predicción de las curvas*, que se pueden comparar con los mismos valores dados al obtener la curva de regresión del modelo único.

2. Y por otra parte se dispone de los valores MMRE y PRED(0,3) obtenidos a partir de las estimaciones del esfuerzo utilizando los proyectos de test, los cuales ofrecen una medida de *predicción del modelo*, que también se pueden comparar con los valores correspondientes del modelo único.

Ambas medidas, a su vez, también pueden ser comparadas y ser objeto de análisis. Las conclusiones que se extraigan de todas estas consideraciones se ofrecerán en las respectivas secciones de cada experimento.

5.6.5. Predicción del esfuerzo que costará llevar a cabo un nuevo proyecto de software

En el caso real de tener que estimar el esfuerzo que tendría que dedicar una empresa a la construcción de un nuevo proyecto de software, disponiendo de una serie de datos de proyectos históricos, se podría afrontar su resolución de dos formas diferentes, e igualmente de distinta efectividad:

1. Para el proyecto del cual se desea conocer su esfuerzo, se escogería la variable o variables conocidas que se utilizaron en el proceso de regresión, a continuación y según el criterio de la sección 5.6.3 se seleccionaría la curva a utilizar, y finalmente a partir de ésta se obtendría el valor del esfuerzo estimado.
   Esta medida del esfuerzo estará sujeta a un grado de incertidumbre, esta medida será del mismo tipo que la asociada al método de la sección 5.6.3. Es decir, los valores MMRE y PRED(0,3) obtenidos mediante la realización de los test allí indicados (los correspondientes a la capacidad predictiva del modelo), serían los mismos aplicables al esfuerzo estimado para este proyecto. Si el valor de predicción del modelo

utilizado, es de PRED(0,3)=24 %, por ejemplo, esa será la probabilidad de que el valor del esfuerzo estimado sea el verdadero.

2. Por otra parte existe la posibilidad de utilizar el proyecto del que se desea conocer el esfuerzo junto con el resto de proyectos en el proceso de segmentación llevado a cabo por la herramienta RCT. Una vez clusterizados los proyectos, es posible, a través de RCT conocer el cluster al que pertenece un determinado proyecto. Nada más fácil pues, averiguar el segmento en el que estaría incluido el proyecto bajo estudio y utilizar la curva de regresión correspondiente a ese segmento para estimar el esfuerzo buscado.

   En este caso, la bondad del esfuerzo estimado vendría dada por el valor de MMRE y PRED(0,3) correspondientes a esa curva de regresión, por ejemplo un valor de PRED(0,3)=62 %, siendo esta medida de bondad, en general, los valores medios de PRED(0,3) y MMRE.

   Para obtener la máxima seguridad en el acierto del esfuerzo estimado, sería posible, gracias a la gran versatibilidad de RCT, realizar sucesivas modelizaciones con diferentes parámetros (diferentes combinaciones de atributos, así como funciones de distribución de probabilidad, regresiones distintas con distintos atributos, etc.) de manera que se utilizaría el proceso, descrito en el primer párrafo de este epígrafe, con el modelo que mejor comportamiento demostrase, consiguiendo así un PRED(0,3) máximo.

   El inconveniente que tendría este método sería el hecho de que el proyecto a predecir formaría parte del proceso de modelización, cosa que iría en contra de los preceptos que se establecen para las técnicas de aprendizaje automático (Mitchell 1997), en las que nunca se utilizan para aprender elementos que más tarde se tomarán como elementos desconocidos a priori para su reconocimiento, o lo que proceda. En este caso concreto de estimación de proyectos de software, y tratándose de un único proyecto de predicción frente a miles de entrenamiento, la influencia sobre el proceso de modelización sería mínima. Además, hay que tener en cuenta que el atributo esfuerzo de este proyecto, aparece vacío, por lo que no se tiene en cuenta

en el proceso de modelización.

El tema de la predicción del esfuerzo para una nueva instancia está abierto a diferentes posibilidades, las cuales serán tratadas convenientemente en el capítulo 6, dedicado a conclusiones y futuras líneas de trabajo.

5.7. Ejecución y análisis de resultados

En esta sección se evalúa la efectividad de la técnica propuesta a través de su aplicación a diversos escenarios de utilización, dando lugar a diferentes experimentos. Las herramientas utilizadas han sido Weka y RCT, construida específicamente para esta tesis, y cuya descripción viene dada en el apéndice A. Weka, se ha utilizado únicamente para probar el algoritmo k-medias, mientras que para probar el algoritmo EM se han utilizado tanto RCT como Weka. Todo el trabajo está basado en la utilización de la herramienta RCT, que implementa el algoritmo EM.

Esta sección se divide, a su vez, en diferentes subsecciones, que recogen todas las pruebas realizadas sobre la base de datos de proyectos ISBSG, filtrada tal y como se estableció en la sección 5.2.

En la sección 5.7.1 se realiza un análisis comparativo de los algoritmos k-medias y EM. A continuación, en la sección 5.7.2 se realiza un experimento comparando, sobre los mismos clusters obtenidos, cada una de las posibles formas de obtener las curvas de regresión que los representan. En la sección 5.7.3 se realiza un análisis recursivo sobre los clusters obtenidos en el experimento anterior, dando lugar a nuevas conclusiones. Finalmente, en la sección 5.7.4 se introduce el conocimiento de expertos, para realizar pruebas que determinen la influencia de ciertos cost drivers sobre los resultados obtenidos.

5.7.1. Comparación de los algoritmos k-medias y EM, con el modelo básico

Estos algoritmos han sido comentados en la sección 5.5, por lo que a continuación se procederá a mostrar los resultados y conclusiones obtenidos tras su utilización en el

proceso de segmentación de la base de datos de proyectos ISBSG. Los proyectos de esta base de datos han sido divididos en dos partes, una de proyectos de entrenamiento, con 1246 proyectos, y otra con proyectos de test, 300 proyectos, siendo éstos los 300 que aparecen en primer lugar en la base de datos original, es decir, no se han elegido de forma aleatoria. Los proyectos de entrenamiento se utilizarán para obtener los segmentos con sus respectivas curvas de regresión, mientras que los proyectos de test, se utilizarán para evaluar el modelo segmentado obtenido.

Las respectivas curvas de regresión de los segmentos determinados por el proceso de agrupamiento, se obtienen utilizando la fórmula 5.3.1, es decir, se utiliza regresión exponencial. La herramienta utilizada para obtener los parámetros A_0 y A_1 de dicha curva, ha sido FuReA[10] (Software Tools for Regression Analysis of Fuzzy Data), la cual permite realizar análisis difuso, así como crisp (que es el utilizado en este caso). Cada curva se ha conseguido utilizando el 70 % de proyectos de cada cluster, mientras que el 30 % restante se ha utilizado para obtener los valores de MMRE y PRED(0,3), los cuales dan una idea de la capacidad predictiva de dicha curva. Se ha utilizado, por lo tanto, cross-validation para evaluar cómo predicen el esfuerzo las curvas obtenidas.

El motivo por el que se eligen los proyectos de entrenamiento de entre los primeros, así como el hecho de usar validación cruzada simple, y usar FuReA (que obtiene las curvas de regresión mediante aproximación exponencial) se debe a que los cálculos se han realizado de forma manual, ya que el algoritmo k-medias no está implementado en RCT, y para que las comparaciones sean congruentes, los experimentos se deben realizar de la misma forma. Más adelante, en los siguientes apartados se verán experimentos totalmente automatizados, y que ofrecen unos resultados sensiblemente diferentes, aunque se utilice la misma base de datos de proyectos.

Para realizar el proceso de segmentación de proyectos, se han elegido dos de los atributos más representativos de cualquier proyecto, el esfuerzo y los puntos de función.

[10]http://fuzzy.ru

K-medias

El algoritmo de k-medias utilizado ha sido el implementado en la herramienta Weka. A este algoritmo hay que suministrarle, además de los proyectos a clusterizar, el número de clusters en los que se desean agrupar los proyectos. Este dato se obtuvo utilizando la herramienta RCT, ya que un paso previo a la clusterización con el algoritmo EM consiste en obtener el número de segmentos óptimo, utilizando v-fold cross validation o el criterio *MDL*, tal como se describió en la sección 5.5.2.

Una vez disponibles, el número de segmentos (8 en este caso) y los proyectos a clusterizar, se puso en funcionamiento el algoritmo k-medias de Weka. De los 19 atributos que describen cada uno de los proyectos, se utilizaron el esfuerzo y los puntos de función para realizar el proceso, mientras que los demás fueron ignorados. Después de 23 iteraciones, los proyectos se agruparon en 8 segmentos, cuyas características se pueden ver en la tabla 5.2. La distribución gráfica de los proyectos en los segmentos obtenidos es la de la figura 5.3.

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
Nº Proyectos	8	848	109	62	163	23	4	31
Media e	17923,75	1661,402	3421,669	21843,507	8753,975	50027,869	128605,25	8241,903
Media fp	12442,875	195,510	1087,650	1150,444	393,869	2644,087	3959,25	3129,258
Desv Stand e	20790,773	1246,735	2115,561	5922,120	2761,958	15483,692	33795,697	6474,577
Desv Stand fp	4400,706	142,453	346,031	529,210	203,18	1368,167	2011,872	1129,781
A_0	896700	130,1	79,75	41520	4976	147300	-	85840000
A_1	-0,4949	0,4445	0,5153	-0,09498	0,09241	-0,1412	-	-1,207
MMRE	-	0,89	1,30	0,22	0,28	0,30	-	1,06
PRED(0,3)	-	29,67 %	32,35 %	61,90 %	58,92 %	57,14 %	-	18,18 %

Tabla 5.2: Resultados del algoritmo k-medias

En la tabla 5.2 también se pueden ver los valores de A_0 y A_1 de las curvas de regresión obtenidas para cada cluster. Para el cluster 6 no se realiza análisis de regresión por estar formado únicamente por 4 proyectos. En el caso del cluster 0, que también está formado por pocos proyectos, sí que se obtiene su curva de regresión, y ello es debido al hecho de que este cluster contiene un conjunto de proyectos muy representativo y diferente al resto, ya que son proyectos mucho mayores que el resto de los presentes en la base de datos (aquellos con valor medio de puntos de función igual a 12442,875), y

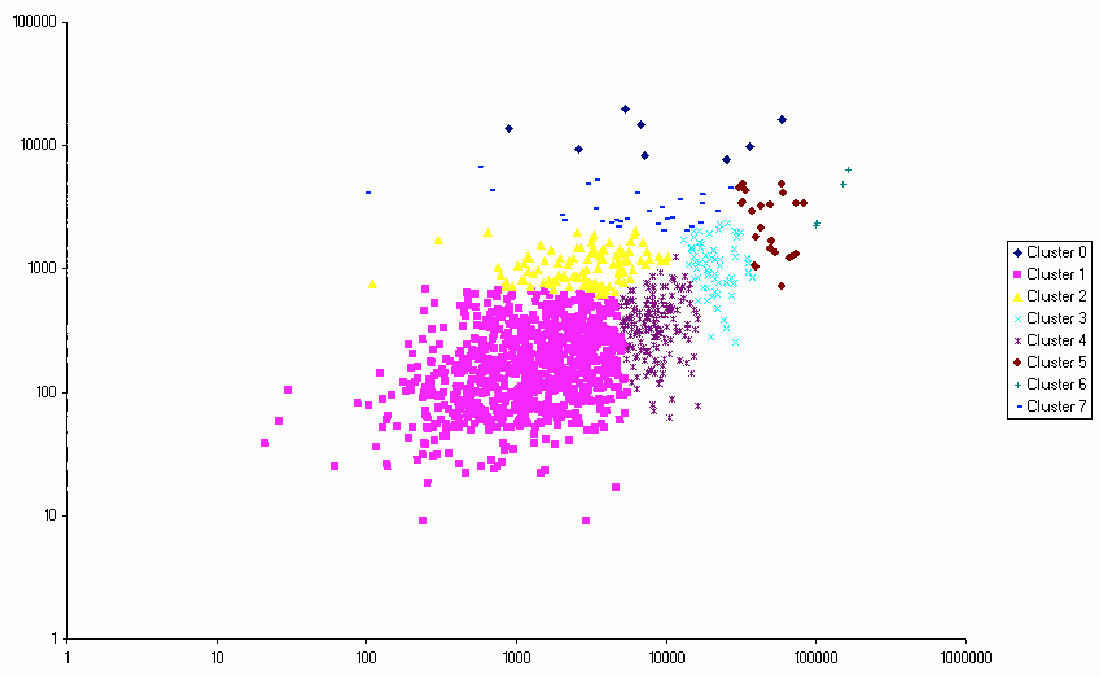

Figura 5.3: Clusters obtenidos con el algoritmo k-medias

es necesario tenerlos en cuenta de cara a la evaluación del modelo de estimación, ya que de otro modo estos proyectos no quedarían representados por el modelo. Con respecto al cluster 6, el valor medio de sus puntos de función (3959,25) es muy parecido al del cluster 7 (3129,25), sin embargo el valor medio del esfuerzo (128605,25) para el cluster 6 es varios órdenes de magnitud mayor al correspondiente para el cluster 7 (8241,903), por lo que no se puede afirmar que el cluster 7 de cobertura a los proyectos del cluster 6, pero finalmente se decide descartarlo por estar formado únicamente por 4 proyectos.

Por último, en este cuadro, también se pueden ver los valores de MMRE y PRED(0,3) de cada una de las curvas de regresión obtenidas, excepto las de los clusters 0 y 6. Como se ha comentado anteriormente, no tiene sentido hacerlo con clusters de tan pocos proyectos.

EM

Para la realización de este experimento se utilizó la herramienta RCT, considerando igualmente los atributos esfuerzo y puntos de función. El no utilizar la herramienta Weka para este fin reside en el hecho de que ésta considera los atributos con los que realiza el proceso de clusterización, independientes entre sí, decisión que en el caso del esfuerzo y los puntos de función no parece demasiado acertada. Éste es uno de los motivos por los que se decidió llevar a cabo la construcción de RCT.

Para obtener el número óptimo de cluster se utilizó el criterio MDL (el mismo que para k-medias), además de seleccionarse los splits iniciales de forma aleatoria. Los atributos utilizados para la segmentación fueron el esfuerzo y los puntos de función, seleccionándose como modelo una distribución normal de estas dos variables. Los resultados obtenidos se pueden ver en la tabla 5.3, así como en la figura 5.4.

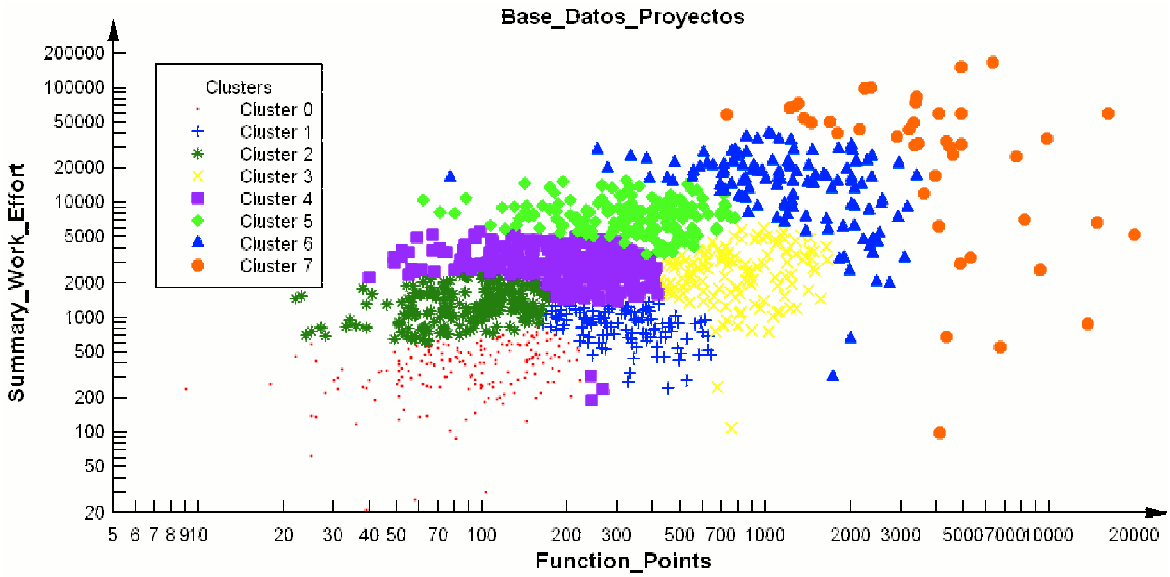

Figura 5.4: Clusters obtenidos con el algoritmo EM

Finalmente se pueden comparar los resultados obtenidos al aplicar estos algoritmos

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
Nº Proyectos	190	117	225	135	254	164	120	42
Probabilidad	0,129	0,085	0,176	0,110	0,205	0,154	0,102	0,036
Media e	410,662	885,072	1259,954	2570,193	2721,609	6946,333	15195,636	41373,482
Media fp	101,562	308,542	96,380	758,840	206,330	366,225	1352,765	4891,646
Desv Stand e	180,148	292,007	557,077	1305,946	1171,040	3171,635	8988,608	36768,719
Desv Stand fp	49,991	139,578	38,915	353,888	93,557	165,120	727,902	4166,534
Coef Correl e-fp	0,3415	-0,3182	0,3015	0,2396	-0,2467	-0,0413	-0,3011	-0,2664
A_0	74,53	6963	372,4	565,5	8570	11540	832600	231600000
A_1	0,3396	-0,3702	0,2708	0,214	-0,2057	-0,07485	-0,5877	-1,158
MMRE	0,42	0,27	0,26	0,92	0,64	0,26	0,93	2,26
PRED(0,3)	35,08 %	71,42 %	64,17 %	52,5 %	53,94 %	71,42 %	36,11 %	8,33 %

Tabla 5.3: Resultados del algoritmo EM

con el modelo paramétrico que utiliza una única curva de regresión para representar a todos los proyectos, cuadro 5.4. Los valores para el modelo único se han obtenido mediante análisis de regresión sobre 1246 proyectos, utilizando FuReA. Los valores de MMRE y PRED(0,3) de este modelo son los resultantes de evaluar esta curva de regresión con los 300 proyectos de test, al igual que para k-medias y EM. Como se puede apreciar, todo se ha realizado sobre los mismos proyectos y con las mismas herramientas, con el objetivo de conseguir realizar unas comparaciones los más uniformes y congruentes posibles.

En la tabla 5.5 se pueden ver los resultados de la comparativa, los valores medios de MMRE y PRED(0,3) de las curvas de los clusters obtenidos, los valores del modelo global, así como los obtenidos con los proyectos de test reservados para la evaluación predictiva del modelo.

	Base Datos Proyectos
Nº Proyectos	1246
A_0	28
A_1	0,806
MMRE	1,92
PRED(0,3)	22,14 %

Tabla 5.4: Análisis de regresión, mediante FuReA, para el modelo paramétrico global

Para el cálculo de estos valores medios no se han tenido en cuenta, para el caso de k-medias, los clusters 0, y 6. La razón, la misma en los dos casos, el número tan escaso de proyectos que los forman, que podrían afectar en gran medida a los resultados obtenidos.

	Predicción Media de las Curvas		Predicción del Modelo	
	MMRE	PRED(0,3)	MMRE	PRED(0,3)
K-medias	0,67	43,02 %	2,88	20,80 %
EM	0,74	49,12 %	2,06	18,79 %
Modelo único	1,92	22,14 %	1,92	22,14 %

Tabla 5.5: Comparación k-medias, EM y modelo paramétrico global

Análisis de resultados y conclusiones

Los valores medios obtenidos mejoran a los que se obtienen con el modelo paramétrico global, siendo los valores obtenidos por el algoritmo EM mejores a los ofrecidos por k-medias. Los segmentos de proyectos obtenidos por EM y k-medias son distintos entre sí, debido a la forma diferente que tienen para la construcción de los clusters. En general los clusters obtenidos por EM tienen un número de proyectos uniforme, salvo excepciones como el cluster 7 que es atípicamente pequeño, sin embargo el algoritmo k-medias da lugar a clusters formados por un número de proyectos muy dispares entre sí, por ejemplo el cluster 1 tiene 848 proyectos, mientras que el cluster 4 tiene 163 y el cluster 7 solamente 31 proyectos. De cara a los valores de predicción medios de MMRE y PRED(0,3) obtenidos, así como a los correspondientes de predicción del modelo, esta característica no influye demasiado, obteniendo en ambos casos, EM y k-medias, valores muy parecidos. Igual ocurre con los valores del modelo global.

A pesar de estas igualdades, EM es un algoritmo más versátil que k-medias y proporciona muchas más posibilidades de cara a la realización de nuevas pruebas sobre la base de datos de proyectos, por ejemplo permite indicar la independencia o dependencia entre los atributos que se utilizarán para segmentar los proyectos. Es por ello por lo que en los próximos experimentos, éste será el algoritmo que se utilizará de cara a la segmentación de proyectos.

En cuanto a los valores en sí que aparecen asociados a MMRE y PRED(0,3) cabría pensar si son todo lo válidos que deberían. La respuesta es no, ya que estos valores se han obtenido realizando una validación cruzada simple, es decir 30 % de proyectos de test y

70 % de proyectos de entrenamiento, lo cual condiciona los resultados a los proyectos de test utilizados, que puede que no sean una muestra todo lo representativa que debiera. Lo ideal sería realizar v-fold cross validation, la cual asegura que los datos de MMRE y PRED(0,3) obtenidos sean mucho más fiables, ya que utiliza todos los datos disponibles para obtener el modelo final. En futuros experimentos será así como se calculen los valores medios de MMRE y PRED(0,3). Los valores de predicción no se pueden calcular mediante v-fold cross validation ya que habría que realizar el proceso de modelado completo (obtener curvas de regresión, valores de A_0 y A_1 de cada curva, realizar la predicción con un conjunto de proyectos de test, y obtener los valores de predicción de MMRE y PRED(0,3)) para diferentes conjuntos de datos de test y entrenamiento, proporcionando diferentes valores, ya no solo de MMRE y PRED(0,3) de predicción, sino también de A_0 y A_1 de cada cluster (los cuales también serían diferentes), luego no se podría arrojar una conclusión media a partir de todos estos resultados.

5.7.2. Experimentación con las diferentes formas de regresión de RCT

El primer experimento que se realizará con la herramienta RCT consistirá en la realización de una clusterización simple sobre 1246 proyectos de entrenamiento elegidos de forma aleatoria, quedando por lo tanto 300 proyectos de test elegidos también de forma aleatoria. Los atributos tomados en consideración serán el esfuerzo y los puntos de función, sobre los que se indicará que siguen una distribución normal bidimensional.

Se trata únicamente de comparar los resultados que se obtienen aplicando las diferentes formas de regresión, con el objetivo de mostrar cuales de ellas ofrecen un mejor comportamiento hacia los datos que forman los proyectos de software.

Tras el primer paso de segmentación sobre los atributos esfuerzo y puntos de función, se obtienen los clusters del cuadro 5.6, indicando en el mismo los datos característicos de cada cluster, como media, desviación típica, etc. La figura 5.5 muestra la representación gráfica de los clusters obtenidos, así como de los elementos de test que se utilizarán para

probar la capacidad predictiva del modelo final obtenido.

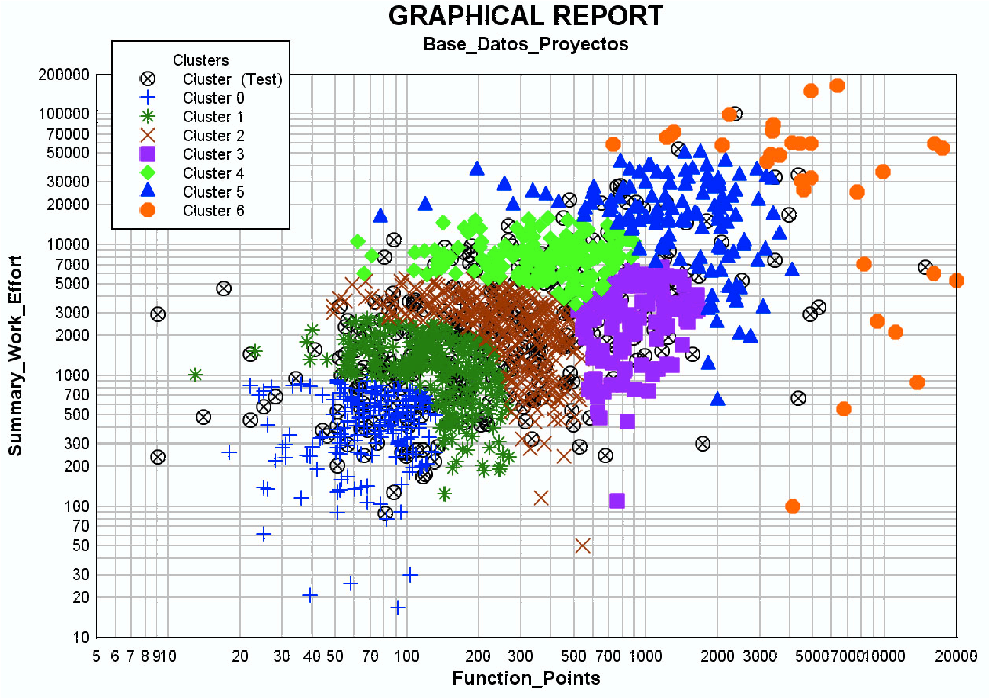

Figura 5.5: Clusters obtenidos con la herramienta RCT

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
Nº Proyectos	177	306	321	114	169	128	31
Probabilidad	0,133	0,235	0,245	0,095	0,157	0,105	0,027
Media e	474,532	1257,227	2533,056	2892,66	7334,681	17849,586	46689,829
Media fp	75,4	135,429	269,177	846,109	406,073	1495,645	6159,518
Desv Stand e	252,233	614,618	1329,888	1646,118	3337,86	11052,253	38191,984
Desv Stand fp	29,352	57,965	119,354	368,016	198,57	801,413	5062,282
Coef Correl e-fp	0,1029	-0,2965	-0,4704	0,31	0,036	-0,1946	-0,3377

Tabla 5.6: Clusters obtenidos por RCT

A continuación se verán en las tablas 5.7, 5.8, 5.9, 5.10 y 5.11 los diferentes valores de los parámetros A_0 y A_1 de las respectivas curvas de regresión obtenidas con las diferentes aproximaciones que permite RCT, así como los valores de MMRE y PRED(0,3) de las mismas. En cada uno de estos cuadros se aprecia, en la primera columna, los valores correspondientes a los 1246 proyectos de entrenamiento sin clusterizar.

Por último se ofrecerá en la tabla 5.12 una comparativa que, por un lado, tiene en

	Proyectos Entrenamiento	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
A_0	1041,36	211,88	1813,016	4168,13	1734,26	7495,35	22657,5	66951,63
A_1	3,606	2,9411	-4,2756	-5,7552	1,0772	0,7585	-4,0293	-3,7789
MMRE	1,68	1	0,53	0,7	0,9	0,26	0,93	21,5
PRED(0,3)	23,86 %	44,06 %	49,01 %	50,46 %	35,96 %	61,53 %	43,75 %	41,93 %

Tabla 5.7: Regresión lineal: *Linear (least-absolute-deviation)*

	Proyectos Entrenamiento	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
A_0	3578,98	421,44	1934,76	4440,94	1362,7	8215,8	24527,98	68741,06
A_1	3,4611	0,5451	-4,6393	-6,5387	1,7117	0,193	-3,6314	-3,056
MMRE	4,11	1,07	0,56	0,69	0,93	0,28	1,13	24,29
PRED(0,3)	14,76 %	40,67 %	47,05 %	51,09 %	35,96 %	60,35 %	35,15 %	38,7 %

Tabla 5.8: Regresión lineal: *Linear (least-squares)*

cuenta los valores predictivos medios de MMRE y PRED(0,3) de cada una de las aproximaciones de regresión dadas, y por otro lado, los correspondientes valores preditivos de MMRE y PRED(0,3) del modelo. Aparecen dichos valores para cada forma de regresión así como para el modelo único.

Para obtener los valores medios de MMRE y PRED(0,3) no se ha tenido en cuenta el cluster 6, ya que para todos los casos ofrece muy malos valores de MMRE, siendo además un cluster con pocos proyectos y dispersos entre sí.

Análisis de resultados y conclusiones

En primer lugar, si se examina el gráfico con los clusters obtenidos, figura 5.5, se observa que el cluster 6 está bastante disperso, los puntos están en un rango de valores de puntos de función de entre 1.000 y 20.000, algo que da idea de la dificultad que se tendrá en su ajuste. Hay otros clusters que muestran igualmente una dispersión, pero menor a éste.

Siguiendo con este cluster 6, y observando las tablas 5.7, 5.8, 5.9, 5.10 y 5.11, se comprueba que los valores de MMRE son excesivamente elevados, siendo los modelos de transformación logarítmica los que mejores valores ofrecen sobre el mismo. Mencionar,

	Proyectos Entrenamiento	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
A_0	38,94	88,23	7783,84	43563,99	245,83	6649,36	128378,68	5763486,49
A_1	0,7343	0,3738	-0,3845	-0,5212	0,3554	0,0287	-0,2899	-0,6075
MMRE	1,58	0,96	0,55	0,74	0,9	0,26	0,95	15,1
PRED(0,3)	22,44 %	45,19 %	48,36 %	46,72 %	37,71 %	61,53 %	37,5 %	32,25 %

Tabla 5.9: Regresión linealizada mediante doble transformación logarítmica: *Linear log-log Plot (least-absolute-deviation)*

	Proyectos Entrenamiento	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
A_0	41	163,88	13797,4	111094,49	19,61	9062,12	248456,92	38312472,85
A_1	0,7174	0,2012	-0,5187	-0,7112	0,7099	-0,0238	-0,3924	-0,8736
MMRE	1,52	0,9	0,51	0,64	0,78	0,26	0,86	12,62
PRED(0,3)	22,97 %	38,41 %	44,44 %	40,8 %	30,7 %	59,17 %	37,5 %	25,8 %

Tabla 5.10: Regresión linealizada mediante doble transformación logarítmica: *Linear log-log Plot (least-squares)*

que para este tipo de clusters con valores tan dispersos, el mejor ajuste es el que lleva a cabo la aproximación *mínima desviación absoluta (least-absolute-deviation)* que le da menor importancia a los puntos muy alejados, por ejemplo en la tabla 5.9 se obtiene una valor 32,25 % para PRED(0,3), mejor que el de 25,8 % obtenido con *mínimos cuadrados (least-squares)*, cuadro 5.10.

En cuanto a los valores de predicción medios, mostrados en la tabla 5.12, destacar que cualquier aproximación de regresión ofrece unos resultados muy parecidos respecto a MMRE y PRED(0,3). Respecto a los valores de predicción del modelo, no hay tampoco mucha diferencia entre el modelo paramétrico segmentado y el modelo que utiliza una única ecuación, pero entre las diferentes formas de regresión sí que hay diferencias, mostrando un mejor comportamiento de predicción la aproximación de transformaciones logarítmicas, en concreto la que utiliza mínima desviación absoluta. Esto puede deberse al hecho de que este criterio, a la hora de ajustar la curva de regresión, funciona muy bien para puntos dispersos, muy abundantes en la base de datos de proyectos ISBSG.

Los valores de predicción medios de las curvas superan con creces a los valores de

	Proyectos Entrenamiento	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
A_0	207,15	316,53	5554,27	29443,88	74,36	8876,21	62584,75	683358,03
A_1	0,5653	0,0892	-0,303	-0,4423	0,5405	-0,0115	-0,1678	-0,3147
MMRE	3,43	1,06	0,59	0,78	0,93	1,07	1,15	22,12
PRED(0,3)	18,05 %	40,67 %	48,69 %	50,77 %	35,08 %	40,82 %	38,28 %	38,7 %

Tabla 5.11: Regresión exponencial: *Non-Linear Power (least-squares)*

	Predicción Media de las Curvas		Predicción del Modelo	
	MMRE	PRED(0,3)	MMRE	PRED(0,3)
RCT. Linear (l.a.d.)	0,72	47,46 %	3,94	21 %
M. Único. Linear (l.a.d.)	1,68	23,86 %	1,36	22 %
RCT. Linear (l.s.)	0,77	45,04 %	7,39	13 %
M. Único. Linear (l.s.)	4,11	14,76 %	3,48	13,34 %
RCT. Linear log-log Plot (l.a.d.)	0,72	46,16 %	32,89	23,33 %
M. Único. Linear log-log Plot (l.a.d.)	1,58	22,44 %	1,41	22,34 %
RCT. Linear log-log Plot (l.s.)	0,65	41,8 %	112,72	22 %
M. Único. Linear log-log Plot (l.s.)	1,52	22,97 %	1,33	23 %
RCT. Non-Linear Power (l.s.)	0,93	42,38 %	11,77	15,66 %
M. Único. Non-Linear Power (l.s.)	3,43	18,05 %	3,09	16 %

Tabla 5.12: Comparación de los valores obtenidos con las diferentes regresiones

predicción del modelo, estos malos resultados no pueden justificarse por una mala elección de los valores de test, ya que como se muestra en la figura 5.5 están bastante bien repartidos entre todo el espacio de proyectos.

Finalmente se puede observar que los valores medios de predicción de las curvas de los modelos segmentados mejoran notablemente a los valores correspondientes obtenidos por el modelo único. Esto permite concluir, que en general, es mejor utilizar las diferentes curvas obtenidas mediante el proceso de segmentación de proyectos, que utilizar una única curva representativa de toda la base de datos de proyectos.

Como último comentario, destacar que cuando se afronte un nuevo problema, lo más conveniente consistirá en utilizar diferentes regresiones, dependiendo de la forma y características de los clusters en cuestión. No es bueno utilizar siempre el mismo tipo de regresión para todos los clusters obtenidos, hay que diversificar, ya que no todos los clusters presentan las mismas características, al igual que las diferentes aproximaciones de regresión. En definitiva, para un mismo experimento, es deseable utilizar diferentes

regresiones con el objetivo de mejorar lo máximo posible.

Después de realizar este análisis, cabe la pregunta, ¿qué se puede hacer ante clusters, como el cluster 6, que están formados por elementos tan dispersos y proporcionan tan malas estimaciones? La respuesta a esta pregunta se verá en el próximo apartado.

Mejora recursiva sobre el cluster 6

Una posibilidad consiste en volver a segmentar este cluster con la intención de formar grupos más homogéneos entre sí. La causa del problema consistía en el mal ajuste de estos puntos con una única curva de regresión, pues veamos que ocurre si se utilizan varias curvas de regresión.

Para este ejemplo se partirá de los resultados obtenidos tras realizar regresión con transformación logarítmica y uso de mínima desviación absoluta, cuadro 5.9.

Si se segmenta a su vez el cluster 6 utilizando igualmente los atributos esfuerzo y puntos de función, siguiendo éstos una distribución normal bivariable, se obtienen los 3 clusters de la figura 5.6, que como se puede observar tienen una mayor cohesión entre sí a nivel de subcluster, aunque siga existiendo dispersión.

Las características de estos clusters son las mostradas en la tabla 5.13.

	Cluster 6-0	Cluster 6-1	Cluster 6-2
Nº Proyectos	7	20	4
Probabilidad	0,223	0,647	0,129
Media e	2530,57	52734,996	107082,793
Media fp	11621,615	3954,917	11237,387
Desv Stand e	2165,792	21923,153	50593,489
Desv Stand fp	5084,393	2342,949	5650,196
Coef Correl e-fp	0,8262	-0,6852	-0,98

Tabla 5.13: Clusters obtenidos a partir del cluster 6

Lo más destacable del cuadro 5.13 son los valores tan grandes de los valores medios tanto del esfuerzo como de los puntos de función, ello se debe a que representan proyectos bastante alejados del resto. Pero lo más interesante reside en el coeficiente de correlación

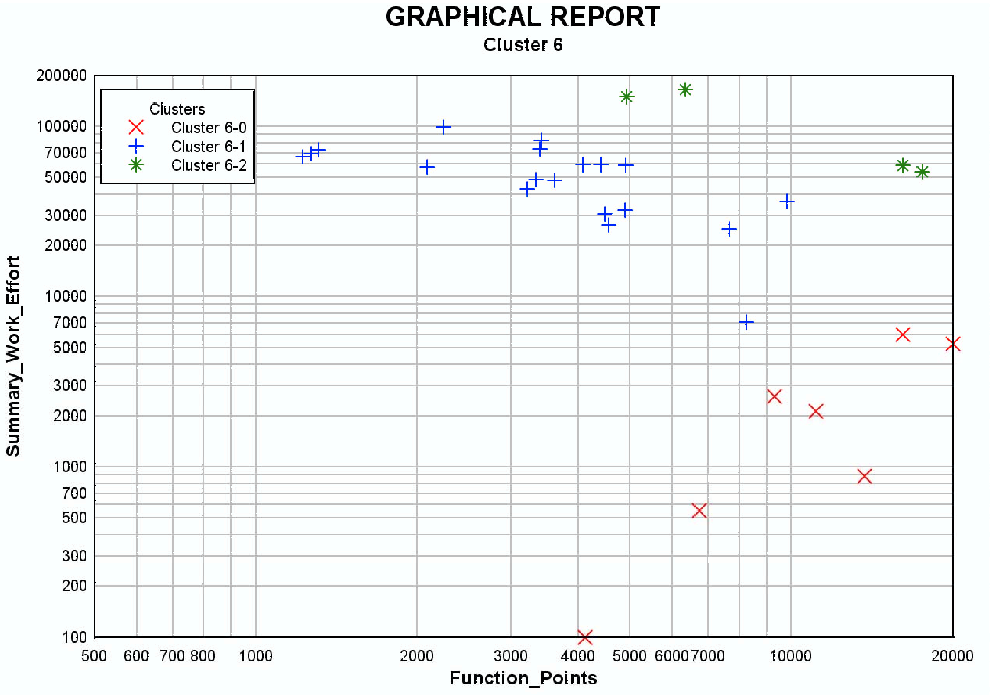

Figura 5.6: Segmentación recursiva del cluster 6

existente entre estas dos variables dentro de cada cluster, denotando estos valores una fuerte dependencia lineal entre ellos, por lo que induce a pensar que una forma adecuada de ajustar estos clusters mediante regresión podría ser a través de regresión lineal. Estos datos se muestran en la tabla 5.14.

	Cluster 6-0 Linear (least-squares)	Cluster 6-1 Linear (least-squares)	Cluster 6-2 Linear (least-absolute-deviation)
A_0	-1555,20	77973,85	187194,58
A_1	0,3517	-6,3540	-7,5667
MMRE	1,23	0,46	0,14
PRED(0,3)	28.57 %	60 %	100 %

Tabla 5.14: Parámetros de las regresiones de los clusters recursivos

Como se puede apreciar en la tabla 5.14 se tiene un valor medio de MMRE=0,61, frente al valor MMRE=15,1 que tenía sin subdividir el cluster 6, cuadro 5.9. Además se tiene el valor medio de PRED(0,3)=62,85 % frente al valor PRED(0,3)=32,25 % sin realizar división recursiva. Y por último el carácter predictivo del modelo pasa de un

MMRE=32,89, cuadro 5.12, a uno de MMRE=3,8, y de una valor PRED(0,3)=23,33 % a un PRED(0,3)=22,6 % (valores obtenidos al realizar un nuevo test considerando los nuevos clusters en los que se divide el cluster 6). Se mejora MMRE pero empeora PRED(0,3), sin embargo los valores medios son mucho mejores.

Se podrían haber elegido otras curvas, pero dada la fuerte relación lineal mostrada a través de los coeficientes de correlación entre el esfuerzo y los puntos de función, parece lógico que sean regresiones lineales, que por otra parte, como se ha podido comprobar ofrecen muy buenos resultados.

Una conclusión muy importante que se puede extraer de estos resultados es que siempre existe posibilidad de mejora, intentando obtener unos clusters (mediante segmentación recursiva, si fuese necesario) con unas formas y características que sean propicias para determinadas formas de regresión. El ejemplo más claro se ha tenido con el cluster 6, el cual estaba formado por un conjunto de proyectos bastante dispersos, y que tras su segmentación recursiva ha dado lugar a clusters con elementos más agrupados y que tras la aplicación de las regresiones más apropiadas ha ofrecido unos resultados mucho más satisfactorios.

5.7.3. Análisis recursivo mediante RCT

El enfoque recursivo es un aspecto muy importante a tener en cuenta de cara a la segmentación de proyectos de software, ya que puede aportar, como se apreció en la sección anterior, para el caso del cluster 6, una mejoría bastante razonable sobre los valores de MMRE y PRED(0,3) de los ancestros de un segmento terminal. Es por ello por lo que en esta sección se procederá a realizar una segmentación recursiva sobre la base de datos de proyectos hasta el nivel de recursión máximo.

El proceso recursivo se realizará sobre la segmentación inicial obtenida en la sección 5.7.2. Todas las divisiones se realizan sobre los mismos atributos, esfuerzo y puntos de función, utilizando igualmente una distribución normal bivariable. Tras realizar todas

las posibles segmentaciones, se obtienen 17 clusters terminales, los cuales se muestran en la figura 5.7.

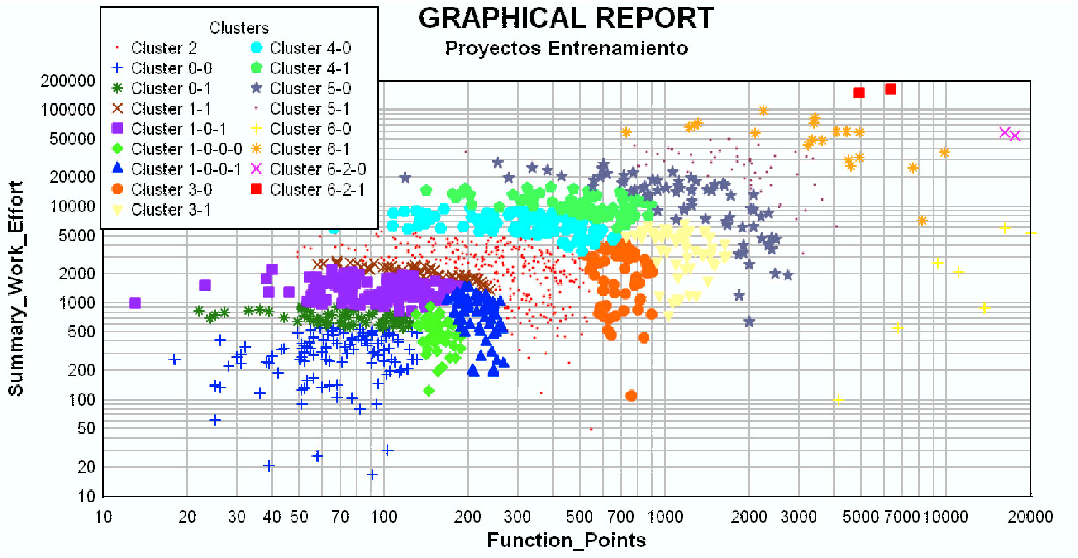

Figura 5.7: Clustering recursivo mediante RCT

A continuación se mostrarán los parámetros de las distintas curvas de regresión obtenidas para cada uno de los clusters según la estructura jerárquica que forman.

Se puede empezar por el cluster 0, el cuál da lugar a otros dos nuevos clusters, el cluster 0-0 y el cluster 0-1. Los resultados se pueden observar en la tabla 5.15. Las características de cada cluster se encuentran en la tabla 5.16. En la figura 5.8 se muestra su distribución en el espacio. Como puede verse los valores medios de MMRE=0,54 y PRED(0,3)=79,46 % de los clusters que se obtienen de forma recursiva, mejoran a los que se obtienen sin esa división, los cuales corresponden al cluster 0, MMRE=0,96 y PRED(0,3)=45,19 %.

El cluster 1 se divide en dos clusters, el cluster 1-0 y el cluster 1-1 (terminal). A su vez, el cluster 1-0 da lugar al cluster 1-0-0 y al 1-0-1 (terminal), produciendo finalmente el cluster 1-0-0 los clusters 1-0-0-0 y 1-0-0-1 (terminales). Los resultados se pueden observar

	Regresión	A0	A1	MMRE	PRED(0,3)
Cluster 0	Linear log-log (l.a.d.)	88,23	0,3738	0,96	45,19 %
Cluster 0-0	Linear log-log (l.a.d.)	91,26	0,3086	0,99	58,92 %
Cluster 0-1	Linear Fitting (l.a.d.)	861,26	-1,8947	0,1	100 %

Tabla 5.15: División recursiva del cluster 0

	Cluster 0-0	Cluster 0-1
Nº Proyectos	112	65
Probabilidad	0,645	0,355
Media e	327,63	707,35
Media fp	73,29	77,08
Desv Stand e	153,928	107,4
Desv Stand fp	27,304	27,97
Coef Correl e-fp	0,2135	-0,4636

Tabla 5.16: Características de los clusters procedentes del cluster 0

en la tabla 5.17. Las características de cada cluster se encuentran en la tabla 5.18. En la figura 5.9 se muestra su distribución en el espacio. Al igual que en el caso anterior los valores medios de MMRE=0,28 y PRED(0,3)=75,57 % de los clusters que se obtienen de forma recursiva, mejoran a los que se obtienen sin esa división, los cuales corresponden al cluster 1, MMRE=0,53 y PRED(0,3)=49,01 %.

	Regresión	A0	A1	MMRE	PRED(0,3)
Cluster 1	Linear Fitting (l.a.d.)	1813,01	-4,2756	0,53	49,01 %
Cluster 1-1	Linear log-log (l.a.d.)	13580,97	-0,3859	0,06	100 %
Cluster 1-0-1	Linear log-log (l.a.d.)	1709,89	-0,0575	0,18	82,99 %
Cluster 1-0-0-0	Linear Fitting (l.s.)	933,93	-2,8576	0,38	65,85 %
Cluster 1-0-0-1	Linear Fitting (l.a.d.)	1975,66	-5,422	0,53	53,44 %

Tabla 5.17: División recursiva del cluster 1

El cluster 2 no se divide de forma recursiva, ya que su forma óptima es la formada por un solo segmento. La información del mismo se puede consultar en la tabla 5.6 y siguientes.

El cluster 3 se divide en dos clusters, el cluster 3-0 y el cluster 3-1 (ambos terminales). Los resultados se pueden observar en la tabla 5.19. Las características de cada cluster se encuentran en la tabla 5.20. En la figura 5.10 se muestra su distribución en el espacio. Al igual que en el caso anterior los valores medios de MMRE=0,8 y PRED(0,3)=51,89 % de los clusters que se obtienen de forma recursiva, mejoran a los que se obtienen sin esa

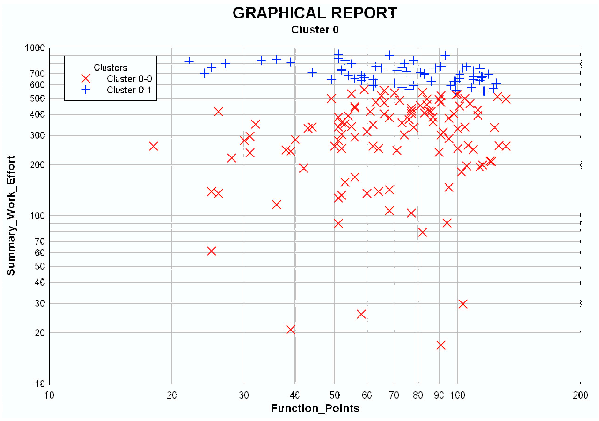

Figura 5.8: Clustering recursivo del cluster 0

	Cluster 1-1	Cluster 1-0-1	Cluster 1-0-0-0	Cluster 1-0-0-1
Nº Proyectos	60	147	41	58
Probabilidad	0,183	0,6	0.41	0.59
Media e	2096,81	1383,05	494,97	787,39
Media fp	141,91	101,15	155,52	215,45
Desv Stand e	337,765	316,107	175,37	328,653
Desv Stand fp	48,28	36,75	14,98	27,304
Coef Correl e-fp	-0,8973	-0,1695	-0,1258	-0,3493

Tabla 5.18: Características de los clusters procedentes del cluster 1

división, los cuales corresponden al cluster 3, MMRE=0,9 y PRED(0,3)=37,71 %.

	Regresión	A0	A1	MMRE	PRED(0,3)
Cluster 3	Linear log-log (l.a.d.)	245,83	0,3554	0,9	37,71 %
Cluster 3-0	Linear log-log (l.a.d.)	9357,67	-0,2223	1	40,32 %
Cluster 3-1	Linear log-log (l.a.d.)	629630,59	-0,7212	0,6	63,46 %

Tabla 5.19: División recursiva del cluster 3

El cluster 4 se divide en dos clusters, el cluster 4-0 y el cluster 4-1 (ambos terminales). Los resultados se pueden observar en la tabla 5.21. Las características de cada cluster se encuentran en la tabla 5.22. En la figura 5.11 se muestra su distribución en el espacio. Al igual que en el caso anterior los valores medios de MMRE=0,16 y PRED(0,3)=88,55 % de los clusters que se obtienen de forma recursiva, mejoran a los que se obtienen sin esa división, los cuales corresponden al cluster 4, MMRE=0,26 y PRED(0,3)=61,53 %.

El cluster 5 se divide en dos clusters, el cluster 5-0 y el cluster 5-1 (ambos terminales).

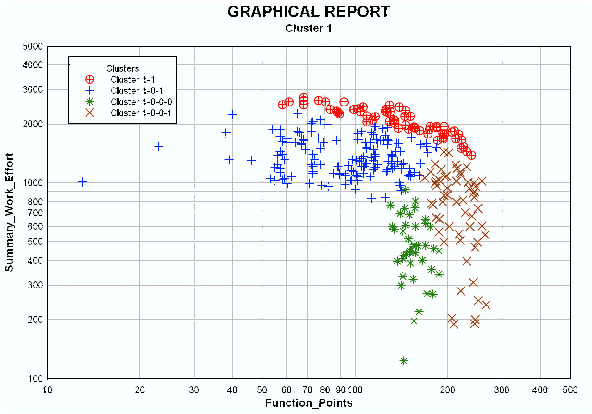

Figura 5.9: Clustering recursivo del cluster 1

	Cluster 3-0	Cluster 3-1
Nº Proyectos	62	52
Probabilidad	0,515	0,485
Media e	2145,39	3736,69
Media fp	696,82	1131,78
Desv Stand e	1154,68	1592,49
Desv Stand fp	110,2	245,59
Coef Correl e-fp	-0,0414	-0,1506

Tabla 5.20: Características de los clusters procedentes del cluster 3

Los resultados se pueden observar en la tabla 5.23. Las características de cada cluster se encuentran en la tabla 5.24. En la figura 5.12 se muestra su distribución en el espacio. Al igual que en el caso anterior los valores medios de MMRE=0,49 y PRED(0,3)=58,02 % de los clusters que se obtienen de forma recursiva, mejoran a los que se obtienen sin esa división, los cuales corresponden al cluster 5, MMRE=0,93 y PRED(0,3)=43,75 %.

El estudio sobre la división del cluster 6 ya se realizó en la sección 5.7.2. El cluster 6 se divide en tres clusters, el cluster 6-0, el cluster 6-1 y el cluster 6-2. El cluster 6-2 está formado por cuatro puntos, los cuales se pueden agrupar en dos clusters de dos puntos más cercanos entre sí, sin embargo a efectos de regresión no tiene sentido hacerlo sobre dos puntos, por lo tanto el cluster 6-2 no se subdividirá en ningún otro. Los resultados se pueden observar en las tablas 5.13 y 5.14. En la figura 5.6 se muestra su distribución en el espacio, teniendo en cuenta que los clusters 6-2-0 y 6-2-1 no se llevan a cabo. Como en todos los casos vistos hasta ahora, los valores medios de MMRE=0,61

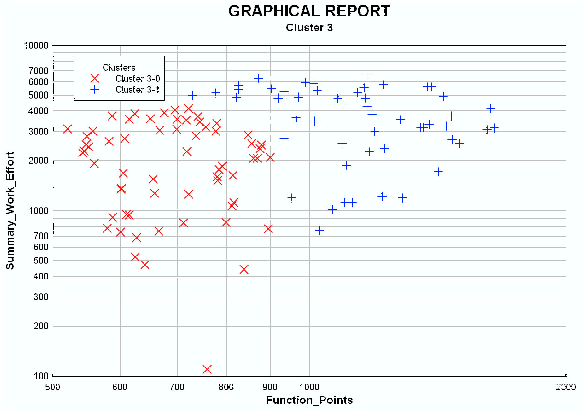

Figura 5.10: Clustering recursivo del cluster 3

	Regresión	A0	A1	MMRE	PRED(0,3)
Cluster 4	Linear log-log (l.a.d.)	6649,36	0,0287	0,26	61,53 %
Cluster 4-0	Linear Fitting (l.a.d.)	8696,44	-5,7705	0,16	89,42 %
Cluster 4-1	Linear log-log (l.s.)	56585,46	-0,2737	0,16	87,69 %

Tabla 5.21: División recursiva del cluster 4

y PRED(0,3)=62,85 % de los clusters que se obtienen de forma recursiva, mejoran a los que se obtienen sin esa división, los cuales corresponden al cluster 6, MMRE=21,5 y PRED(0,3)=41,93 %.

Los valores medios de MMRE y PRED(0,3) obtenidos por los clusters terminales del proceso de recursión se comparan con los mejores valores de MMRE y PRED(0,3) del modelo total. A partir del cuadro 5.12 se observa cuál es la mejor regresión, en este caso la lineal, de la cuál se extraen los valores de MMRE y PRED(0,3) para el modelo único. Igualmente se pueden comparar estos valores con los de MMRE y PRED(0,3) del caso no recursivo, elegidos también como los más altos del cuadro 5.12, que también proceden de la regresión lineal. La exponencial con doble transformación logarítmica ofrece unos resultados muy parecidos, por lo que no habría que descartarla, ya que muy probablemente, si se hubiese elegido un conjunto de proyectos de entrenamiento distinto (recuérdese que la elección es aleatoria), los resultados podrían ser muy diferentes. En la tabla 5.25 se puede apreciar la mencionada comparación.

	Cluster 4-0	Cluster 4-1
Nº Proyectos	104	65
Probabilidad	0,588	0,412
Media e	6780,38	10453,7
Media fp	337,41	505,4
Desv Stand e	1512	2605,98
Desv Stand fp	145,19	199,44
Coef Correl e-fp	-0,4794	-0,3769

Tabla 5.22: Características de los clusters procedentes del cluster 4

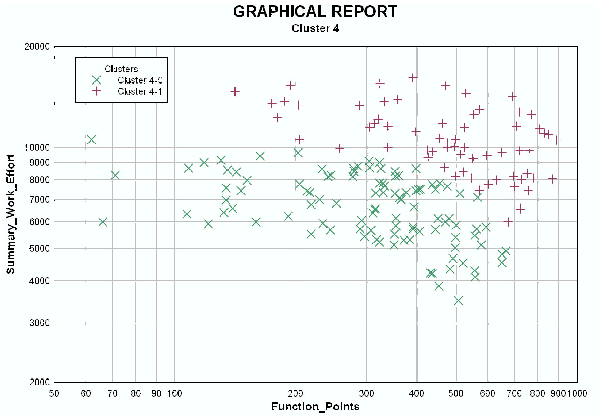

Figura 5.11: Clustering recursivo del cluster 4

Análisis de resultados y conclusiones

Como puede observarse en los resultados ofrecidos, los clusters que muestran a través de su coeficiente de correlación una dependencia lineal fuerte o moderada, la utilización de una regresión lineal suele ofrecer mejores resultados, mientras que para el resto de casos la regresión exponencial linealizada, mediante doble transformación logarítmica, es la que mejores resultados ofrece. Así mismo, para clusters con proyectos dispersos funciona mejor la aproximación mediante mínima desviación absoluta (*least absolute deviation*), y para los que no, la aproximación más utilizada de mínimos cuadrados (*least squares*).

Si se examina con detenimiento la tabla 5.25 se puede observar un par de aspectos interesantes:

	Regresión	A0	A1	MMRE	PRED(0,3)
Cluster 5	Linear Fitting (l.a.d.)	22657,5	-4,0293	0,93	43,75 %
Cluster 5-0	Linear Fitting (l.s.)	24556,97	-8,7224	0,53	61,33 %
Cluster 5-1	Linear log-log (l.s.)	1344191,79	-0,5403	0,46	54,71 %

Tabla 5.23: División recursiva del cluster 5

	Cluster 5-0	Cluster 5-1
Nº Proyectos	75	53
Probabilidad	0,561	0,439
Media e	13309,03	25866,99
Media fp	1330,16	1880,37
Desv Stand e	7022,51	11503,23
Desv Stand fp	673,41	841,03
Coef Correl e-fp	-0,7878	-0,4552

Tabla 5.24: Características de los clusters procedentes del cluster 5

- En la columna *Predicción del Modelo* aparecen los valores de MMRE y PRED(0,3), obtenidos a partir de la evaluación de los modelos bajo estudio, mediante la utilización de proyectos de test. Los valores de MMRE y PRED(0,3) del modelo que utiliza una única función paramétrica, son mejores que los que se obtienen mediante el modelo segmentado, ya sea recursivo o no. Las diferencias en cuanto a PRED(0,3) no son muy considerables (en torno al 1 % - 2 %), siendo mucho mayores en cuanto a MMRE, indicando que los desvíos de los valores estimados con respecto a los reales son verdaderamente grandes.

- Los valores de predicción medios (columna *Predicción Media de las Curvas*) de MMRE y PRED(0,3) de los modelos segmentados, superan con creces a los correspondientes del modelo único. Además el modelo recursivo ofrece mejores valores de estos índices que el no recursivo, un 20 % mayor en cuanto a PRED(0,3) y dos décimas mejor que MMRE.

  El proceso recursivo mejora al normal ya que los clusters que se van obteniendo en las sucesivas segmentaciones están formados por elementos más homogéneos entre sí, de manera que las regresiones correspondientes proporcionan unas curvas mucho más ajustadas a los elementos de dichos clusters.

Se puede concluir que siempre es deseable llevar a cabo el análisis recursivo cuando se

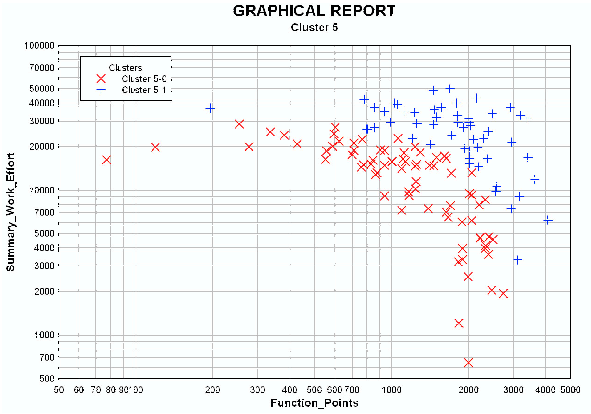

Figura 5.12: Clustering recursivo del cluster 5

	Predicción Media de las Curvas		Predicción del Modelo	
	MMRE	PRED(0,3)	MMRE	PRED(0,3)
Modelo Recursivo	0,51	66,77 %	4,36	19,33 %
Modelo No Recursivo	0,72	47,46 %	3,94	21 %
Modelo Único (Linear Fitting (l.a.d.))	1,68	23,86 %	1,36	22 %

Tabla 5.25: Comparación del proceso recursivo con el modelo no recursivo y con el total

quiera un mejor ajuste de las curvas de regresión y unos valores de MMRE y PRED(0,3) más cercanos a lo deseable.

5.7.4. Particionado basado en el conocimiento de expertos y posterior clusterización automática de proyectos

En esta sección se va a seguir un procedimiento similar al seguido en el trabajo de J.J. Cuadrado et al. (Cuadrado et al. 2006) para la segmentación de proyectos de software. Se va ha utilizar el conocimiento previo que se dispone sobre ciertos cost drivers para realizar particiones de los proyectos de la base de datos de proyectos ISBSG. En concreto se utilizarán 1246 proyectos de entrenamiento para obtener el modelo y 300 proyectos de test para evaluarlo.

Del proceso de particionado resultarán nueve segmentos, los cuales a su vez serán sometidos a sendos procesos de clusterización. Sobre los clusters finales es sobre los que se realizará el posterior análisis. Y todo ello mediante la utilización de la herramienta RCT.

Proceso de particionado

En el trabajo de J.J. Cuadrado (Cuadrado et al. 2006) quedó demostrada la influencia que tienen ciertos cost drivers en el desarrollo de proyectos y por lo tanto en la formación de segmentos. Dos de los atributos más relevantes en este sentido son, el uso de alguna metodología de diseño de software en el proyecto (METHO), atributo "Used Methodology", y la utilización de herramientas CASE (CASET), atributo "Upper CASE Used".

Si se particionan los proyectos según los posibles valores de cada uno de estos atributos METHO y CASET ("yes/no/unkown")[11], se obtienen nueve particiones, correspondientes a las nueve posibles combinaciones de valores ("yes/yes" - cluster#7, "yes/no" - cluster#8, "yes/unknown" - cluster#9, "no/yes" - cluster#10, "no/no" - cluster#11, "no/unknown" - cluster#12, "unknown/yes" - cluster#34, "unknown/no" - cluster#35, "unknown/unknown" - cluster#36), los cuales se muestran en la figura 5.13.

Proceso de clusterización

Para cada una de estas nueve particiones se realiza el proceso de clusterización correspondiente mediante el algoritmo EM, teniendo en cuenta para la realización del mismo los atributos esfuerzo y puntos de función.

La jerarquía de clusters obtenidos se puede ver en la figura 5.14.

Para la clusterización de las particiones obtenidas se utilizan, como se ha mencionado anteriormente los atributos esfuerzo y puntos de función, modelizados mediante una función normal bivariable, excepto para la partición correspondiente al cluster#10, para el que se ha utilizado una distribución t-Student de estas dos variables, debido a la naturaleza de los datos de este cluster#10, los cuales presentan un alto grado de dispersión.

En la figura 5.15 se pueden observar los cinco clusters obtenidos tras segmentar el cluster#7, que contiene proyectos que han sido creados utilizando metodologías de diseño y herramientas CASE.

[11]Se utilizarán los términos en inglés.

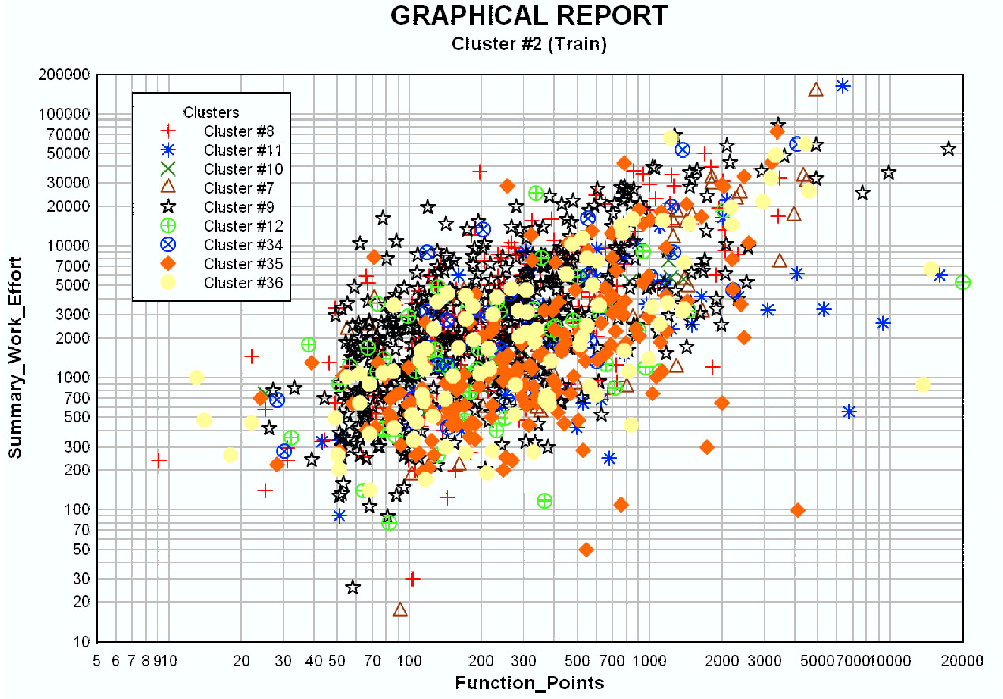

Figura 5.13: Particionado de la base de datos ISBSG según los atributos METHO y CASET

Las características correspondientes a los clusters obtenidos a partir de la partición cluster#7 son las que aparecen en la tabla 5.26.

	Clust#13	Clust#14	Clust#15	Clust#16	Clust#17
Nº Proy	19	7	18	13	5
Probabilidad	0,272	0,111	0,309	0,226	0,08
Media e	750,23	1865,5	4189,08	14606,01	47824,39
Media fp	251,7	176,77	395,56	1538,75	4236,59
Desv Std e	445,45	487,92	2259,69	10048,05	51983,39
Desv Std fp	128,71	92,31	217,92	403,44	488,47
Coef Correl e-fp	0,5612	-0,9703	0,0144	0,6521	0,8077

Tabla 5.26: Clusters obtenidos a partir del cluster#7

Los parámetros de las regresiones obtenidas para cada uno de estos clusters son los que figuran en la tabla 5.27.

Figura 5.14: Jerarquía de clusters obtenidos tras el proceso de particionado y posterior clusterización

En la figura 5.16 se pueden observar los tres clusters obtenidos tras segmentar el cluster#8, que contiene proyectos que han sido creados utilizando metodologías de diseño en su construcción pero no usan herramientas CASE.

Las características correspondientes a los clusters obtenidos a partir de la partición cluster#8 son las que aparecen en la tabla 5.28.

Los parámetros de las regresiones obtenidas para cada uno de estos clusters son los que figuran en la tabla 5.29.

En la figura 5.17 se pueden observar los cinco clusters obtenidos tras segmentar el cluster#9, que contiene proyectos que han sido creados utilizando metodologías de diseño en su construcción pero se desconoce si usan herramientas CASE.

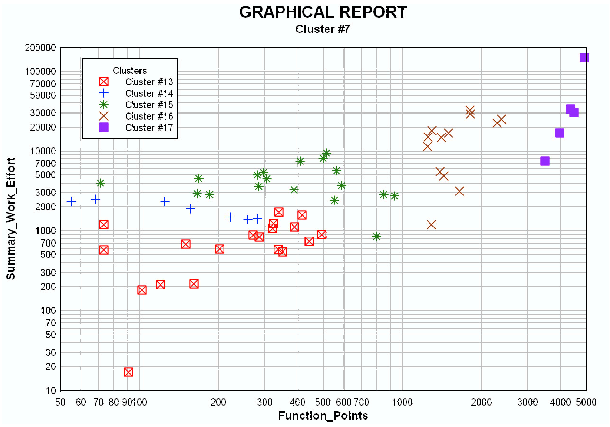

Figura 5.15: Segmentación del cluster#7, METHO yes, CASET yes

	Regresión	A0	A1	MMRE	PRED(0,3)
Cluster#7	Linear Fitting (l.a.d.)	-503,43	7,9293	1,63	16,12 %
Cluster#13	Linear log-log (l.a.d.)	3,48	0,9701	2,46	31,57 %
Cluster#14	Linear Fitting (l.s.)	2782,82	-5,2095	0,08	100 %
Cluster#15	Linear Fitting (l.a.d.)	5035,73	-2,4577	0,46	50 %
Cluster#16	Linear log-log (l.a.d.)	9,04	1,0219	1,58	46,15 %
Cluster#17	Linear Fitting (l.a.d.)	10^{-7}	6,6354	0,26	60 %

Tabla 5.27: Características de los clusters procedentes de la partición cluster#7

Las características correspondientes a los clusters obtenidos a partir de la partición cluster#9 son las que aparecen en la tabla 5.30.

Los parámetros de la curva de regresión obtenida para cada uno de estos clusters son los que figuran en la tabla 5.31.

	Clust#18	Clust#19	Clust#20
Nº Proy	116	76	30
Probabilidad	0,509	0,347	0,142
Media e	1825,44	5728,77	20770,33
Media fp	142,39	382,13	1400,66
Desv Std e	1166,38	3710,18	11851,93
Desv Std fp	67,08	203	761,72
Coef Correl e-fp	0,5253	-0,0117	-0,0323

Tabla 5.28: Clusters obtenidos a partir del cluster#8

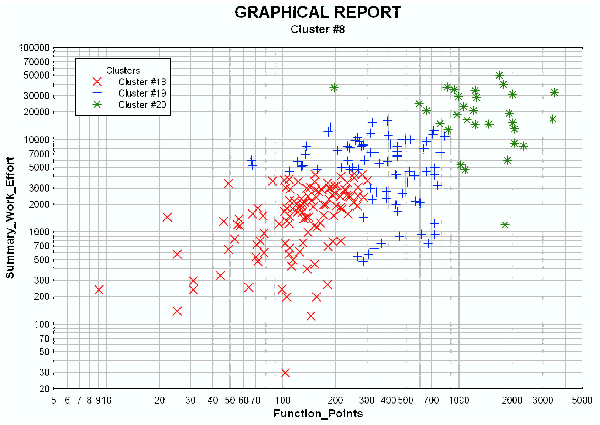

Figura 5.16: Segmentación del cluster#8, METHO yes, CASET no

	Regresión	A0	A1	MMRE	PRED(0,3)
Cluster#8	Linear Fitting (l.a.d.)	756,71	9,2524	1,42	29,27 %
Cluster#18	Non Linear Power Fitting	66,38	0,6741	1,35	50 %
Cluster#19	Non Linear Power Fitting	10123,92	-0,0897	1,42	36,84 %
Cluster#20	Linear log-log (l.s.)	134117,89	-0,2879	0,98	40 %

Tabla 5.29: Características de los clusters procedentes de la partición cluster#8

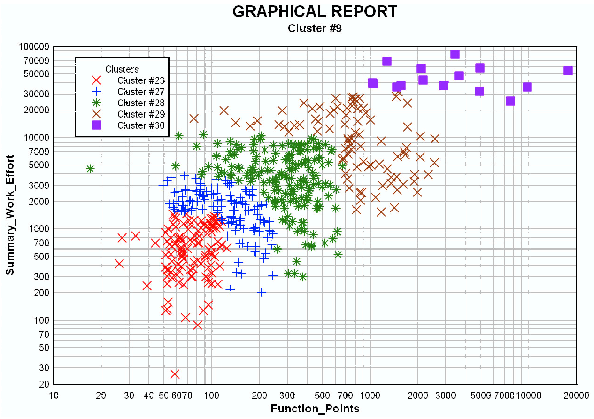

Figura 5.17: Segmentación del cluster#9, METHO yes, CASET unknown

	Clust#23	Clust#27	Clust#28	Clust#29	Clust#30
Nº Proy	119	123	180	88	15
Probabilidad	0,216	0,228	0,347	0,179	0,028
Media e	715,54	1784,61	4170,77	11828,24	45880,16
Media fp	76,76	128,85	304,44	912,15	4366,26
Desv Std e	406,21	856,9	2551,76	8355,48	15744,52
Desv Std fp	22,27	53,11	138,8	558,1	4284,69
Coef Correl e-fp	0,2411	-0,4015	-0,144	-0,2635	0,0013

Tabla 5.30: Clusters obtenidos a partir del cluster#9

	Regresión	A0	A1	MMRE	PRED(0,3)
Cluster#9	Linear log-log (l.a.d.)	33,16	0,7983	1,11	24,38 %
Cluster#23	Linear log-log (l.s.)	67,11	0,5013	0,8	29,41 %
Cluster#27	Linear log-log (l.s.)	47356,87	-0,7126	0,46	52,84 %
Cluster#28	Linear log-log (l.s.)	38694,97	-0,4276	0,78	35 %
Cluster#29	Linear log-log (l.s.)	255758,19	-0,5	0,73	31,81 %
Cluster#30	Linear log-log (l.s.)	55505,07	-0,0277	0,3	66,67 %

Tabla 5.31: Características de los clusters procedentes de la partición cluster#9

En la figura 5.18 se pueden observar los dos clusters obtenidos tras segmentar el cluster#10, que contiene proyectos que no usan metodologías de diseño en su construcción pero que han sido creados utilizando herramientas CASE. Para la segmentación de este cluster se ha utilizado una distribución t-Student con 5 grados de libertad, la cual tiene una amplia cola (pesada) respecto a la distribución normal, o dicho de otra forma, es adecuada para datos dispersos.

Las características correspondientes a los clusters obtenidos a partir de la partición cluster#10 son las que aparecen en la tabla 5.32.

	Clust#21	Clust#22
Nº Proy	8	5
Probabilidad	0,62	0,38
Media e	875,86	5644,86
Media fp	152,73	981,85
Desv Std e	403,48	1305,79
Desv Std fp	94,56	496,45
Coef Correl e-fp	0,7829	0,7

Tabla 5.32: Clusters obtenidos a partir del cluster#10

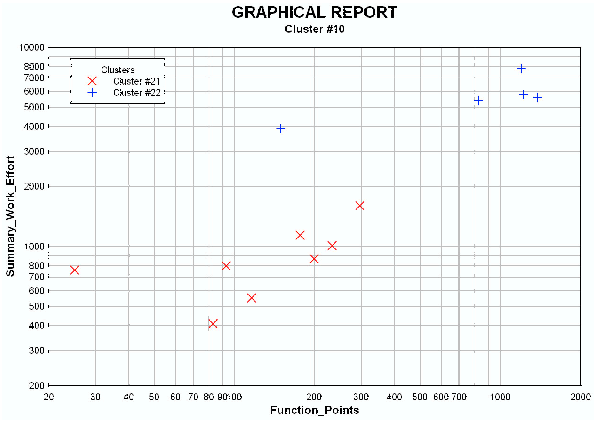

Figura 5.18: Segmentación del cluster#10, METHO no, CASET yes

	Regresión	A0	A1	MMRE	PRED(0,3)
Cluster#10	Linear Fitting (l.a.d.)	272,22	4,5235	0,39	38,46 %
Cluster#21	Non Linear Power Fitting	58,94	0,5467	0,3	62,5 %
Cluster#22	Linear Fitting (l.a.d.)	3659,77	1,7531	0,15	100 %

Tabla 5.33: Características de los clusters procedentes de la partición cluster#10

Los parámetros de las regresiones obtenidas para cada uno de estos clusters son los que figuran en la tabla 5.33.

En la figura 5.19 se pueden observar los tres clusters obtenidos tras segmentar el cluster#11, que contiene proyectos que no usan metodologías de diseño en su construcción ni herramientas CASE.

Las características correspondientes a los clusters obtenidos a partir de la partición cluster#11 son las que aparecen en la tabla 5.34.

Los parámetros de las regresiones obtenidas para cada uno de estos clusters son los que figuran en la tabla 5.35.

En la figura 5.20 se pueden observar los tres clusters obtenidos tras segmentar el

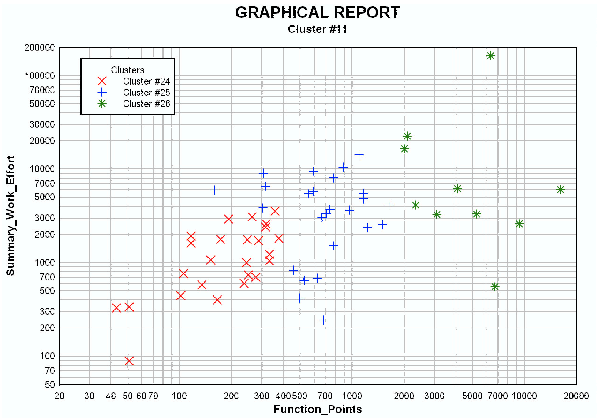

Figura 5.19: Segmentación del cluster#11, METHO no, CASET no

	Clust#24	Clust#25	Clust#26
Nº Proy	25	25	10
Probabilidad	0,4	0,431	0,168
Media e	1378,12	4524,98	22929,04
Media fp	207,6	740,64	5686,56
Desv Std e	959,73	3402,13	47518,75
Desv Std fp	100,1	383,82	4147,39
Coef Correl e-fp	0,5415	0,0696	-0,0004

Tabla 5.34: Clusters obtenidos a partir del cluster#11

	Regresión	A0	A1	MMRE	PRED(0,3)
Cluster#11	Linear Fitting (l.a.d.)	2087,44	0,2445	1,65	21,67 %
Cluster#24	Linear log-log (l.s.)	6,85	0,968	0,66	32 %
Cluster#25	Linear Fitting (l.s.)	4460,4	0,2802	2,27	32 %
Cluster#26	Linear Fitting (l.a.d.)	3679,83	0,154	1,37	22,22 %

Tabla 5.35: Características de los clusters procedentes de la partición cluster#11

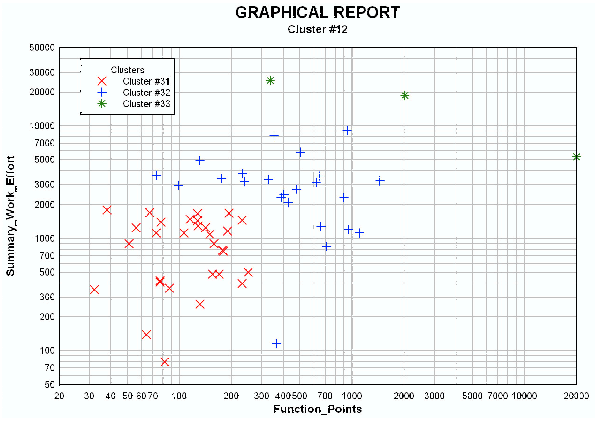

Figura 5.20: Segmentación del cluster#12, METHO no, CASET unknown

	Clust#31	Clust#32	Clust#33
Nº Proy	30	23	3
Probabilidad	0,517	0,429	0,053
Media e	935,96	3171,57	16485,99
Media fp	123,6	512,18	7451,99
Desv Std e	513,86	2111,45	8358,08
Desv Std fp	57,77	345,98	8899,43
Coef Correl e-fp	0,0067	-0,0501	-0,968

Tabla 5.36: Clusters obtenidos a partir del cluster#12

cluster#12, que contiene proyectos que no usan metodologías de diseño en su construcción y se desconoce si utilizan herramientas CASE.

Las características correspondientes a los clusters obtenidos a partir de la partición cluster#12 son las que aparecen en la tabla 5.36.

Los parámetros de las regresiones obtenidas para cada uno de estos clusters son los que figuran en la tabla 5.37.

En la figura 5.21 se pueden observar los dos clusters obtenidos tras segmentar el cluster#34, que contiene proyectos de los cuales se desconoce si han utilizado metodologías de diseño en su construcción pero que han sido creados utilizando herramientas CASE.

	Regresión	A0	A1	MMRE	PRED(0,3)
Cluster#12	Linear Fitting (l.a.d.)	1392,42	0,1954	1,62	26,78 %
Cluster#31	Linear Fitting (l.s.)	940,16	-0,0063	1,21	36,67 %
Cluster#32	Linear log-log (l.a.d.)	10674,43	-0,2204	1,49	43,47 %
Cluster#33	Linear log-log (l.a.d.)	235147,08	-0,3827	0,1	66,67 %

Tabla 5.37: Características de los clusters procedentes de la partición cluster#12

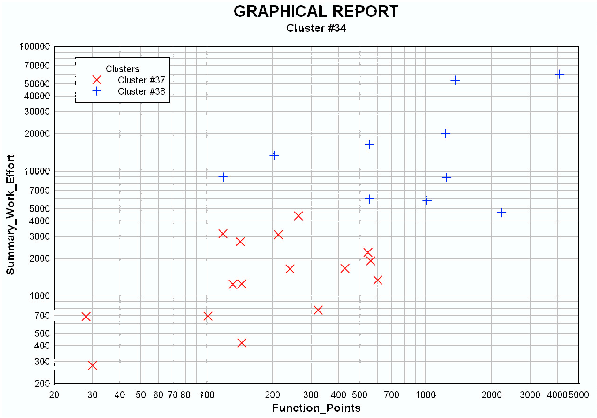

Figura 5.21: Segmentación del cluster#34, METHO unknown, CASET yes

Las características correspondientes a los clusters obtenidos a partir de la partición cluster#34 son las que aparecen en la tabla 5.38.

Los parámetros de las regresiones obtenidas para cada uno de estos clusters son los que figuran en la tabla 5.39.

En la figura 5.22 se pueden observar los tres clusters obtenidos tras segmentar el

	Clust#37	Clust#38
Nº Proy	16	10
Probabilidad	0,61	0,39
Media e	1752,94	19499,23
Media fp	251	1238,52
Desv Std e	1167,32	19141,53
Desv Std fp	182,34	1110,97
Coef Correl e-fp	0,2278	0,6487

Tabla 5.38: Clusters obtenidos a partir del cluster#34

	Regresión	A0	A1	MMRE	PRED(0,3)
Cluster#34	Linear Fitting (l.a.d.)	-818,62	14,88	1,48	19,23 %
Cluster#37	Linear log-log (l.a.d.)	198,53	0,372	0,59	43,75 %
Cluster#38	Linear log-log (l.a.d.)	1794,96	0,3387	1,24	30 %

Tabla 5.39: Características de los clusters procedentes de la partición cluster#34

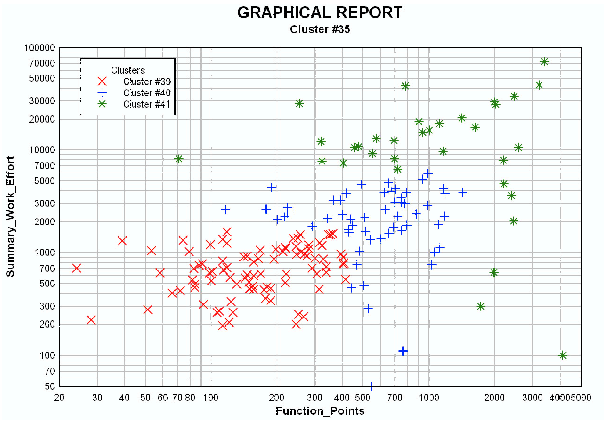

Figura 5.22: Segmentación del cluster#35, METHO unknown, CASET no

cluster#35, que contiene proyectos de los cuales se desconoce si han utilizado metodologías de diseño en su construcción y no han sido creados utilizando herramientas CASE.

Las características correspondientes a los clusters obtenidos a partir de la partición cluster#35 son las que aparecen en la tabla 5.40.

Los parámetros de las regresiones obtenidas para cada uno de estos clusters son los que figuran en la tabla 5.41.

	Clust#39	Clust#40	Clust#41
Nº Proy	91	57	33
Probabilidad	0,475	0,329	0,194
Media e	750,02	2326,78	15274,86
Media fp	186,42	589,76	1398,92
Desv Std e	362,87	1331,76	14931,23
Desv Std fp	101,59	299,98	994,19
Coef Correl e-fp	0,3298	0,2229	0,2693

Tabla 5.40: Clusters obtenidos a partir del cluster#35

	Regresión	A0	A1	MMRE	PRED(0,3)
Cluster#35	Linear Fitting (l.a.d.)	200,18	3,5454	2,19	30,38 %
Cluster#39	Linear log-log (l.s.)	196,43	0,2417	0,49	43,95 %
Cluster#40	Linear log-log (l.s.)	1836,86	0,008	1,6	38,59 %
Cluster#41	Linear Fitting (l.s.)	-3614,24	1329,5391	0,57	37,03 %

Tabla 5.41: Características de los clusters procedentes de la partición cluster#35

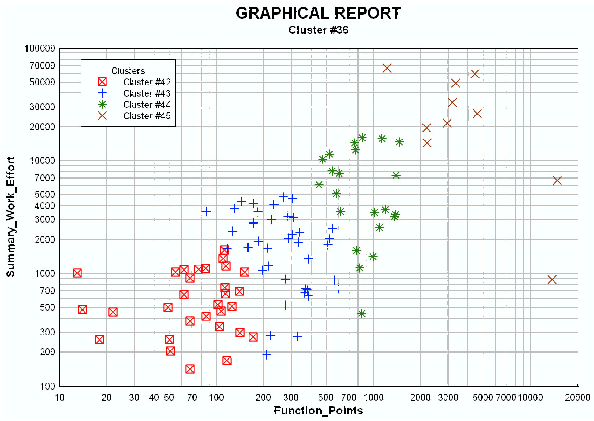

Figura 5.23: Segmentación del cluster#36, METHO unknown, CASET unknown

En la figura 5.23 se pueden observar los cuatro clusters obtenidos tras segmentar el cluster#36, que contiene proyectos de los cuales se desconoce si han utilizado tanto metodologías de diseño en su construcción como herramientas CASE.

Las características correspondientes a los clusters obtenidos a partir de la partición cluster#36 son las que aparecen en la tabla 5.42.

	Clust#42	Clust#43	Clust#44	Clust#45
Nº Proy	30	39	22	10
Probabilidad	0,302	0,359	0,238	0,099
Media e	677,24	2139,46	6567,31	29750,11
Media fp	89,91	273,07	853,29	5223,94
Desv Std e	402,47	1326,78	5067,72	21021,3
Desv Std fp	46,43	123,09	333,12	4596,86
Coef Correl e-fp	0,1542	-0,3301	0,0614	-0,595

Tabla 5.42: Clusters obtenidos a partir del cluster#36

Los parámetros de las regresiones obtenidas para cada uno de estos clusters son los que figuran en la tabla 5.43.

	Regresión	A0	A1	MMRE	PRED(0,3)
Cluster#36	Linear Fitting (l.a.d.) (l.a.d.)	975,98	1,71	2,06	25,74 %
Cluster#42	Linear log-log (l.s.)	380,81	0,0857	0,67	33,34 %
Cluster#43	Linear Fitting (l.a.d.)	3341,87	-4,3309	1,19	35,89 %
Cluster#44	Linear log-log (l.a.d.)	4628676,27	-1,007	1,16	31,94 %
Cluster#45	Linear log-log (l.s.)	381531340,28	-1,1993	1,8	30 %

Tabla 5.43: Características de los clusters procedentes de la partición cluster#36

Finalmente, en la tabla 5.44 se muestran los valores de predicción del modelo. Para tener una visión más amplia se añade como información la ofrecida en la tabla 5.25, con el fin de ver mejor los diferentes resultados obtenidos.

Análisis de resultados y conclusiones

En primer lugar destacar el hecho de la utilización de todos los proyectos en los que aparecen los atributos METHO y CASET con valores desconocidos. Si no fuese así quedarían sin representación cinco de las nueve posibles particiones realizadas, perdiéndose por lo tanto precisión en la forma de los clusters obtenidos, llevando asociado a ello unos resultados de MMRE y PRED(0,3) de menor precisión.

De esta forma, todos los casos están contemplados, teniéndose un cluster para cada uno de ellos, dando cabida a cualquier posibilidad en cuanto a los valores de METHO y CASET se refiere.

	Predicción Media de las Curvas		Predicción del Modelo	
	MMRE	PRED(0,3)	MMRE	PRED(0,3)
Modelo Particionado - Clusterizado	0,95	45,01 %	1,63	30,76 %
Modelo Único (Linear Fitting (l.a.d.))	1,68	23,86 %	1,36	22 %
Modelo No Recursivo	0,72	47,46 %	3,94	21 %
Modelo Recursivo	0,51	66,77 %	4,36	19,33 %

Tabla 5.44: Modelo particionado y posteriormente clusterizado

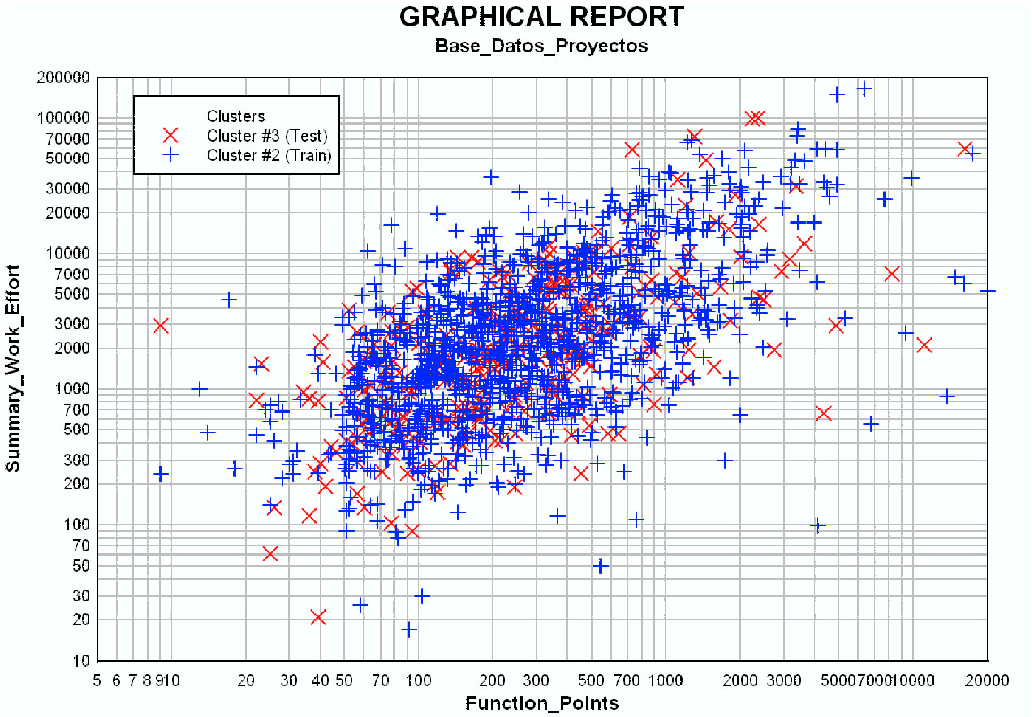

Figura 5.24: Distribución de los proyectos de test sobre los de entrenamiento

Esta es una de las razones por las que, a la hora de realizar los ensayos con los proyectos de test, los resultados obtenidos para MMRE y PRED(0,3) (cuadro 5.44) son tan satisfactorios, ya que entre los valores de test se encuentran muchos proyectos con valores nulos ya sea para el atributo METHO, o para el CASET, o ambos, que se asignan de forma correcta. En la figura 5.24 se puede apreciar la distribución de los valores de test, y como se puede observar, ésta es uniforme con respecto al resto de proyectos de entrenamiento, es decir, que es una muestra muy representativa de los datos que se utilizan en la modelización.

Respecto a la forma en la que se ha realizado el experimento, destacar dos aspectos:

- La distribución normal n-dimensional es la que mejores clusters obtiene, y por lo tanto, ha sido la más ampliamente utilizada. También se ha querido probar la utilización de la distribución t-Student con proyectos dispersos en el espacio, dando también muy buen resultado para el caso del cluster #10 (figura 5.18 y cuadros

5.32 y 5.33).

- Como se ha podido observar, en los resultados mostrados se han utilizado indistintamente cualquiera de los diferentes tipos de aproximaciones para la obtención de las respectivas curvas de regresión que proporciona RCT. Eso sí, se ha tratado de utilizar la más acorde con las características del segmento en cuestión, sobre todo en lo referente a su forma y al valor del coeficiente de correlación entre el esfuerzo y los puntos de función del mismo.

Otro aspecto a mencionar, y que corrobora lo ya comentado y demostrado con resultados en secciones anteriores, pero que aquí se vuelve a recalcar, es que si se observan las tablas que ofrecen los valores de MMRE y PRED(0,3) (cuadro 5.27 y sucesivos), los valores de los segmentos obtenidos mejoran significativamente los valores correspondientes al cluster del que proceden. En definitiva, la división de un conjunto de proyectos en varios subconjuntos (segmentación de proyectos) mejora el proceso de estimación del esfuerzo de proyectos de software.

El proceso de particionado previo a la clusterización hace que los clusters obtenidos sean diferentes a los que se daría lugar si se prescindiese de esa partición inicial. Por ejemplo, existen clusters como el #14, #18 y #27, que tienen valores medios de esfuerzo y puntos de función muy parecidos entre sí. Si no fuese por la partición inicial en base a los atributos METHO y CASET, es muy probable que estos clusters formasen un único cluster. Lo mismo se puede apreciar para los clusters #15, #19 y #28.

Este hecho, es un factor adicional que favorece la asignación de proyectos a clusters, dando lugar tal y como se aprecia en la tabla 5.44 a unos valores de predicción del modelo muy buenos.

Finalmente, si se estudia detenidamente la tabla 5.44 se observa que los mejores valores de predicción del modelo corresponden al presente caso, presentando un valor de PRED(0,3)=30,76 %, superando en más de 8 puntos al caso más próximo. El valor de MMRE mejora también significativamente a los obtenidos en los experimentos de las

secciones anteriores, estando a la par con respecto al modelo único.

En cuanto a los valores de predicción medios de las curvas de regresión, únicamente el caso recursivo mejora a éste, sobre el que también se podría aplicar dicha aproximación y obtener sin duda unos resultados superiores. Los resultados de MMRE y PRED(0,3) presentados son también bastante aceptables y mejores que los del modelo único.

5.8. Resumen

En este capítulo se ha evaluado el modelo paramétrico segmentado mediante la realización varios experimentos sobre la misma base de datos de proyectos y con diferentes aproximaciones. El trabajo realizado permite examinar la consecución de los objetivos planteados en el primer capítulo, actividad que se abordará en el capítulo siguiente.

Capítulo 6

Conclusiones y trabajo futuro

La recompensa del trabajo bien hecho es la oportunidad de hacer más trabajo bien hecho.

Jonas Edward Salk

Una vez presentado el modelo paramétrico segmentado, así como su evaluación, tan sólo resta extraer y analizar las conclusiones y realizar una síntesis del trabajo realizado.

El resto de este capítulo presenta, en primer lugar, un resumen de los objetivos y su evaluación. A continuación, se proporcionan las conclusiones que se han obtenido de la investigación y el trabajo realizado en esta tesis. Después, se resaltan las aportaciones originales que han surgido del proceso de investigación. Finalmente, se presentan las líneas de investigación que quedan abiertas en el campo, y otras que pueden constituir la continuación del esfuerzo realizado.

6.1. Resumen de objetivos y de su evaluación

En esta sección se describe cómo se han alcanzado tanto el objetivo general enunciado en el primer capítulo, como los objetivos específicos, indicando en ambos casos el proceso de evaluación realizado. Este proceso de evaluación permitirá indicar el grado de cumplimiento de los mismos.

6.1.1. Resumen del objetivo general

La finalidad fundamental de esta tesis es hacer una aportación significativa al área de estimación en Ingeniería del Software, y más concretamente a la estimación del esfuerzo necesario en la construcción de un proyecto de software. Para ello se ha definido un nuevo modelo de estimación consistente en la utilización conjunta de técnicas de Inteligencia Artificial con las ya clásicas fórmulas paramétricas de Ingeniería del Software. Se han utilizado algoritmos de clustering conocidos, con el fin de obtener grupos de proyectos sobre los que aplicar diferentes funciones paramétricas, con la misma forma pero con diferentes parámetros.

Todo lo anterior resalta las características novedosas que el presente trabajo supone con respecto al estado de la cuestión, y postula el modelo propuesto como modelo de referencia para su utilización dentro de la herramienta RCT, con la intención de proporcionar a las empresas de la industria software una guía a seguir en el proceso de estimación del coste de futuros proyectos.

6.1.2. Objetivos específicos y su evaluación

La primera actividad que se llevó a cabo consistió en el cumplimiento del objetivo cuarto *"Comparar los resultados obtenidos aplicando diferentes algoritmos de clustering en la generación de modelos paramétricos segmentados"*. Existen diferentes algoritmos de clustering y era necesario determinar cuál o cuales eran los más adecuados para tratar el tipo de datos en cuestión (proyectos de software). El objetivo consistía en seleccionar sin ambigüedad únicamente un algoritmo de clustering, para utilizarlo en el resto del trabajo.

Se realizaron pruebas de segmentación utilizando diversos algoritmos de clustering, procedentes de diferentes familias, consistiendo la comparación de los mismos en el estudio de los valores MMRE y PRED(0,3) obtenidos a partir de cada uno de los segmentos de proyectos dados por el proceso de clusterización sobre la base de datos de proyectos inicial.

Entre los resultados obtenidos y las características de los algoritmos de clustering,

aparecía como más apropiado para los datos bajo estudio, el algoritmo EM, el cuál sería el utilizado en el resto del presente trabajo.

Para cumplir con el resto de objetivos, se procedió a trabajar sobre el segundo de ellos *"Diseñar un algoritmo genérico para la generación de modelos paramétricos segmentados de Ingeniería del Software"*. Este segundo objetivo dio lugar a la construcción de una herramienta integrada, que permite de forma recursiva, si se desea, obtener el modelo paramétrico segmentado de una base de datos de proyectos de software, y realizar sobre el mismo diferentes análisis numéricos y gráficos. Por lo tanto la consecución de este objetivo ha quedado plenamente conseguido a través de esta herramienta, RCT.

Los objetivos primero (*"Comparar las propiedades predictivas de los modelos paramétricos segmentados con modelos paramétricos no segmentados mediante el uso de datos empíricos"*) y tercero (*"Estudiar la influencia de la utilización de conductores de coste conocidos, en la generación de modelos paramétricos segmentados de Ingeniería del Software"*), se han conseguido tras la realización de numerosos experimentos, algunos de los cuales se han mostrado en el capítulo anterior, mediante la utilización de RCT.

En cuanto a la comparación de los modelos paramétricos con los modelos paramétricos segmentados, ha quedado patente la mejoría que se obtiene mediante la utilización de estos últimos. Al igual ha ocurrido con los experimentos realizados en los que se ha utilizado el conocimiento de expertos, mostrando un comportamiento superior a los modelos tradicionales. Estos resultados se pueden consultar el el capítulo anterior.

6.2. Conclusiones

Cada uno de los experimentos realizados ha aportado las conclusiones propias del mismo. En esta sección se ofrecerán unas conclusiones generales a partir de todas las ofrecidas en los respectivos apartados del capítulo anterior.

- La primera y más importante conclusión que se puede postular es que para realizar la estimación del coste del software, disponiendo de una base de datos de proyectos históricos *heterogéneos*, es más adecuado utilizar múltiples funciones de estimación, correspondientes cada una de ellas a cada uno de los segmentos obtenidos tras un proceso de segmentación de los datos, que utilizar una única fórmula que represente a la totalidad de los datos.

  Esta hipótesis se ha visto confirmada tras la realización de múltiples pruebas. Las medidas de comparación utilizadas, MMRE y PRED(0,3), han mostrado unos mejores valores para el modelo segmentado que para el no segmentado. Para los valores de predicción medios de las curvas de regresión queda suficientemente claro, para el caso de los valores de predicción del modelo, únicamente cuando se produce aportación del conocimiento de expertos se mejora al modelo único, aunque en el resto de casos los valores son muy parecidos.

- Otra conclusión importante que se puede extraer de todo lo realizado es que los resultados obtenidos realmente no son lo más importante. Lo verdaderamente importante reside en la utilidad de la metodología seguida, que debido a la gran cantidad de aspectos configurables que conlleva permite realizar multitud de aproximaciones diferentes sobre el mismo objetivo. Se pueden obtener mejores o peores resultados, pero lo que está garantizado es que es posible encontrar los buenos, ya que esta metodología lo ha demostrado.

- El análisis recursivo de los datos ha mostrado que los segmentos obtenidos de esta forma disfrutan de una mayor cohesión, y por lo tanto la función de estimación que se utiliza para ellos ofrece unos resultados más acertados, ya que para su obtención se ha dispuesto de datos más homogéneos.

- La utilización de diferentes aproximaciones para obtener las correspondientes rectas de regresión, así como la modelización de los datos mediante diferentes distribuciones de probabilidad, ha tenido el objeto de representar a segmentos de diferentes formas y características, lo que ha contribuido a que los resultados finales obtenidos hayan sido mucho más fiables.

- Si se dispone de información previa, tal como cierto conocimiento de expertos, es posible, antes del proceso de segmentación, realizar un particionado de los datos basándose en dicho conocimiento. Esto hará que los datos de partida, antes de su clusterización, muestren unas características bastante más homogéneas que si no se hiciese ninguna partición inicial. La experimentación ha demostrado que este procedimiento mejora significativamente los valores de MMRE y PRED(0,3) obtenidos.

Además hay que tener en cuenta la base de datos de proyectos sobre los que se ha realizado la evaluación. Esta base de datos, a pesar de ofrecer una ventaja al estar formada por multitud de proyectos con muchos atributos, tiene un inconveniente que consiste en la gran diversidad de fuentes en cuanto a la procedencia de los datos, teniéndose proyectos muy diferentes entre sí y algunas veces con muchos atributos desconocidos. Otra base de datos que no tuviese este inconveniente seguramente proporcionaría aún mejores resultados.

6.3. Aportaciones originales

La aportación fundamental de la tesis es un nuevo modelo de estimación software que combina técnicas de Inteligencia Artificial, tales como algoritmos de clustering, con técnicas de Ingeniería del Software, tales como la utilización de ecuaciones paramétricas o el conocimiento de expertos.

Las aportaciones concretas de esta tesis son las siguientes:

- Construcción de una herramienta integrada de estimación software RCT. Esta herramienta parte de cualquier base de datos de proyectos y es capaz de proporcionar un modelo de estimación para cualquier atributo deseado.
- Comparación del comportamiento de los modelos paramétricos frente a los modelos paramétricos segmentados. Mostrando estos últimos una mejor conducta de cara a la estimación de cualquier atributo de un proyecto de software, en concreto del esfuerzo.

Entre las aportaciones adicionales significativas que surgen como resultado del proceso de investigación, es posible destacar las siguientes:

- Como consecuencia de la construcción de RCT, es posible la modelización de los datos mediante diferentes distribuciones de probabilidad, así como el ajuste de los mismos mediante distintas formas de regresión. La herramienta realiza análisis local y global de los resultados que se van obteniendo. Y por último permite representar gráficamente los proyectos deseados. Todo esto proporciona tal potencia, que la prueba de todas las combinaciones posibles podría dar unos resultados sorprendentes.

- La idea de seguir segmentando la base de datos de proyectos de forma recursiva, aporta mayor riqueza al modelo final obtenido.

- La aportación de conocimiento ofrecido por expertos humanos, para la mejora en el proceso de estimación, no es algo nuevo, de hecho ha sido ampliamente estudiado por diferentes autores, tales como Boehm (Boehm 1981). Pero la combinación del mismo junto con un proceso de segmentación mediante algoritmos de clustering, supone una novedad en el campo de la estimación del coste del software.

Por último, cabe destacar que durante el proceso de elaboración de esta tesis se han originado trabajos de investigación que han servido para la consecución de los objetivos previstos en el proyecto inicial. Estos trabajos son los siguientes:

- Recursive segmentation of software projects for the estimation of development effort (Garre et al. 2004). En el cual se aborda la aproximación recursiva como forma de mejorar el ajuste de los segmentos obtenidos.

- Segmented Parametric Software Estimation Models: Using the EM algorithm with the ISBSG 8 database (Garre et al. 2005). Muestra el uso del algoritmo EM sobre un ejemplo particular de la base de datos de proyectos ISBSG.

- Estimación del esfuerzo de un proyecto de software utilizando el criterio MDL-EM y componentes normales N-Dimensionales. Aplicación a un caso práctico (Garre &

Charro 2005). Afronta el problema de la estimación introduciendo nuevos conceptos, tales como el criterio MDL-EM para la formación de los segmentos. Así mismo se muestra su aplicación a un nuevo caso.

- An empirical study of process-related attributes in segmented software cost-estimation relationships (Cuadrado et al. 2006). Demuestra la utilidad de la introducción del conocimiento de expertos en el proceso de segmentación de proyectos.
- Comparación de diferentes algoritmos de clustering en la estimación del coste en el desarrollo de software (Garre et al. 2006). Elabora una comparativa entre diferentes algoritmos de clustering para conocer cuál o cuales de ellos ofrecen mejor comportamiento de cara a la segmentación de proyectos de software.

Las aportaciones enumeradas en este apartado no constituyen una lista de resultados cerrada, y pueden servir, por tanto, como punto de partida para otras aplicaciones que utilicen la metodología propuesta.

6.4. Líneas de trabajo futuras

Tanto a lo largo del desarrollo de esta tesis, como a la finalización de la misma, se han observado posibles líneas de trabajo futuras que incluyen por un lado, extensiones a los experimentos realizados, y por otra inclusión de nuevas características que mejoren el proceso de estimación. En esta sección se describen las más relevantes, así como otras no incluidas en esta clasificación.

6.4.1. Extensiones a los experimentos realizados

En los experimentos realizados en el capítulo anterior se han explicitado una serie de resultados. Lo importante realmente no son los resultados obtenidos, sino el establecimiento de un método de trabajo para la obtención de fórmulas de evaluación de costes de proyectos. Es tal la cantidad de posibilidades que ofrece RCT, que es necesario seguir utilizándola con diferentes parámetros, de manera que se utilice todo su potencial con el fin de obtener incluso mejores resultados que los aquí mostrados.

Son varias las posibilidades que se ofrecen y que pueden introducir cambios en las estimaciones realizadas:

Parámetros de la clusterización. Como principal aspecto con el que se puede ensayar es con la forma compuesta de la función de densidad de probabilidad que el algoritmo EM utiliza para representar a los proyectos de la base de datos (*Finite Mixture Model*). Atributos que siguen una distribución junto con otros que siguen otra diferente, etc. Otro aspecto importante del proceso de clusterización lo determinan los *Parámetros adicionales*, tales como el número de splits para la validación cruzada, la selección aleatoria o no de los mismos, o si se utiliza o no validación cruzada frente al criterio MDL, como forma de validar el proceso del algoritmo EM.

Curvas de regresión. Es posible elegir diferentes combinaciones de variable dependiente e independiente, al igual que entre diferentes aproximaciones de regresión, tales como la lineal o de doble transformación logarítmica, por ejemplo. Así mismo, para una regresión dada, es posible cambiar el número de splits que se utilizarán para realizar v-fold cross validation.

Particionado de datos. Son muchas también las combinaciones posibles a la hora de particionar los proyectos: antes de clusterizar, una vez clusterizados, para todos los clusters, o solo para algunos, etc.

Aplicación recursiva. Se puede descender de forma recursiva por una rama o a lo ancho de todo el árbol, hasta el nivel que se desee.

Según se configuren todas estas posibilidades de una forma u otra, los resultados obtenidos serán diferentes. Puede que una de estas combinaciones sea la que mejor resultado ofrezca, aunque no se haya encontrado hasta el momento.

6.4.2. Introducción de nuevas características

En este momento la herramienta RCT está en un estado muy avanzado de desarrollo, pero sin embargo carece de algunas funcionalidades que deberán de introducirse en un futuro próximo, algunas de ellas son:

1. Respecto a las funciones de densidad de probabilidad disponibles, para el tipo de datos que se manejan (proyectos software de la base de datos ISBSG), son suficientes y cubren con bastante acierto las características de los mismos. Sin embargo si se desea que esta herramienta sea general para cualquier fuente de datos, habría que introducir nuevas funciones de probabilidad. Mención especial cabe hacer sobre los datos categóricos, a los cuales es necesario dedicarles mayor atención.

2. La regresión utilizada es bivariable, es decir, se supone que la relación existente entre proyectos se establece en base a la relación lineal o de otro tipo existente entre solamente dos de sus atributos. Esto no es real en la mayoría de los casos, y una variable no está relacionada única y exclusivamente con otra, sino que, en general, se ve afectada por múltiples factores. Es necesario por lo tanto introducir un modelo de regresión multivariable, en el que se pueda obtener una curva de regresión que ajuste una serie de puntos en los que el esfuerzo dependa no solamente de los puntos de función, sino también del uso de una metodología de diseño, por ejemplo, o de cualquier otro atributo.

3. Poca complejidad requeriría el introducir la segmentación recursiva automática, llegando al nivel óptimo de división. Actualmente hay que hacerlo de forma manual.

4. Por último, ¿se podría mejorar la forma de predecir el esfuerzo de un nuevo proyecto?

Estimación del esfuerzo de un nuevo proyecto

En el capítulo anterior ya se describieron dos formas de obtener la estimación del esfuerzo de un proyecto que se desea construir. Una de ellas consistía en utilizar el criterio

de la distancia de Mahalanobis, mientras que la otra consistía en la utilización del nuevo proyecto junto con los de entrenamiento, para obtener el modelo y a partir del mismo determinar el cluster al que pertenece, y por lo tanto la función de estimación a utilizar.

Existe otra posibilidad que se debería tener en cuenta como trabajo futuro y consiste en olvidarse de las curvas de regresión y utilizar directamente las características que el algoritmo EM ofrece respecto a la clusterización. Este algoritmo, al igual que otros, permite tratar atributos con valor desconocido a priori (*missing attributes*), de manera que además de proporcionar una segmentación final, es capaz de ofrecer un valor estimado para esos atributos inicialmente desconocidos. Esto forma parte del funcionamiento del algoritmo EM, que inicialmente le da un valor al atributo desconocido, y de forma iterativa y según el criterio de maximización del log-likelihood, lo va modificando hasta que al final del proceso, el valor que se tenga para ese atributo será el más probable de acuerdo a este criterio y al cluster al que pertenece.

Por lo tanto para realizar la estimación de un atributo no sería necesario nada más que ofrecer el valor final que para este atributo, inicialmente desconocido, calcula el algoritmo EM en su proceso de clusterización.

6.4.3. Otras líneas de trabajo futuras

Se recogen aquí los aspectos no mencionados en las secciones anteriores:

- Considerar otras medidas de evaluación del modelo paramétrico segmentado, diferentes a MMRE y PRED(l), así como utilizarlas de diferentes maneras, tal y como se mencionó en el capítulo 4. Por ejemplo se podría tener en cuenta el tamaño de los clusters, ya que todos tienen el mismo peso a la hora de calcular los valores medios, pero sin embargo quizás no debería ser así, ya que no todos tienen el mismo número de elementos. Además, en cuanto a los elementos de un mismo cluster, habría que descartar a los outliers, de manera que no influyesen negativamente en las medidas de evaluación de dicho cluster.

- Realizar un análisis cualitativo de los resultados obtenidos, atendiendo así a las

características de los diferentes segmentos del modelo, desde el punto de vista de sus cost drivers. Es decir, sería interesante descubrir, si existe, una regla que identificase a los proyectos de un determinado segmento, a través de los valores de algunos de sus atributos. Esta información permitiría afrontar de una forma más eficaz, nuevos procesos de clusterización, y el análisis de los mismos.

- Consideración de datos "fronterizos". Con este nombre se pretende resaltar el hecho de que muchos proyectos se encuentran "cercanos" a diferentes segmentos (no solamente a uno), sin embargo los algoritmos de clustering utilizados, terminan asignando cada uno de los proyectos a un único segmento. Sería interesante tener en cuenta el aspecto "difuso" de la pertenencia de proyectos que, podrían pertenecer con diferentes valores de su función de pertenencia, a distintos segmentos a la vez. De esta forma, a la hora de estimar el esfuerzo de un proyecto, se tendría una combinación ponderada de las diferentes ecuaciones de los segmentos a los que puede pertenecer el proyecto en consideración.

Apéndice A

Recursive Clustering Tool - RCT

En este capítulo se describe el funcionamiento y la forma de utilización de la herramienta de clusterización, diseñada y construida para la realización de todos los experimentos que han hecho posible el desarrollo de esta tesis. Para ello, en primer lugar se hará una descripción de la interface de usuario, seguido de una descripción de una sesión cualquiera de trabajo con RCT, para terminar con una sección en la que se mostrará la potencia de RCT a través de todas las posibles combinaciones de sus opciones.

Es necesario recordar que el algoritmo de clustering utilizado por la herramienta es el algoritmo EM, cuya descripción se llevó a cabo en el capítulo 5.

A.1. Descripción de la interface de usuario

El área visual de RCT se divide en tres ventanas:

Ventana principal. En esta ventana se visualiza toda la información del proceso realizado. Está formada por diversas pestañas, que permiten acceder a diferentes vistas de esta información:

Main Tree. Visualiza, en forma de árbol, los clusters y subclusters obtenidos tras el proceso de clusterización, el cual puede ser recursivo. El nodo raíz representa la base de datos total, de la que cuelgan el resto de nodos que representan los clusters obtenidos. Si se utilizan tanto conjunto de entrenamiento como

conjunto de test, éstos también serán visualizados en esta estructura jerárquica, tal y como aparece en la figura A.4.

Cluster Report. Ofrece información acerca del proceso de clusterización realizado para un conjunto de proyectos dado, el cual puede constituir la base de datos completa, o simplemente un cluster obtenido previamente. Indica el número de clusters obtenidos, y para cada uno de ellos, su probabilidad, media y la desviación típica de cada uno de los atributos utilizados en la segmentación, y los valores de los coeficientes de correlación entre estos atributos.

Regression Report. Esta pestaña se puede aplicar a cada cluster, así como a la base de datos completa, y muestra después de haber realizado el correspondiente análisis, la curva de regresión que lo define. En detalle, los valores que se ofrece son los correspondientes a MMRE y PRED(l) para esa curva, así como los valores de A_0 y A_1 que se han obtenido para el ajuste de la misma. Además se indica el tipo de regresión utilizado en el proceso.

Full Test Report. En el caso de que se haya realizado un test de los resultados obtenidos tras la clusterización, esta pestaña muestra información completa acerca de ese test. Para ello se debe estar posicionado sobre el nodo raíz. La información que se obtiene aparece agrupada en tres apartados:

Model Definition. Descripción de las regiones obtenidas para poder asignar las instancias de test. En definitiva la definición del modelo.

Cluster Link Details. Se ofrecen los valores numéricos de las medias, varianzas y covarianzas utilizadas en el apartado anterior para describir las regiones.

Test Result. En el que aparecen el número de instancias de test utilizadas y los valores de MMRE y PRED(l) obtenidos que dan idea de la capacidad predictiva real del modelo construido.

Graphical Report. A partir de un nodo del árbol de clusters, se muestran gráficamente todos los proyectos de cada uno de los clusters que están bajo éste,

y sólo los expandidos en el árbol (podría haber algún cluster subdividido, pero no expandido, que se mostraría gráficamente en su conjunto, sin mostrar los clusters en los que se divide). El gráfico puede ser exportado a la herramienta gráfica DPlot Jr[1], de libre distribución, con la que se pueden realizar operaciones más sofisticadas sobre el mismo.

Data Bar. Muestra los datos de los proyectos del cluster, o de toda la base de datos de proyectos, seleccionado en la ventana principal. Además aparece el número de proyectos de ese elemento.

Control Bar. Mustra diferentes mensajes de los procesos seguidos, tanto de clusterización como de análisis.

Existen además dos barras más:

Barra de Herramientas. Contiene una serie de iconos con los que se puede acceder de forma rápida a las funcionalidades ofrecidas a su vez, por las opciones de la barra de menú.

Barra de Estado. Visualiza información acerca de las diferentes opciones sobre las que se posiciona el ratón.

Y por supuesto las diferentes opciones de menú:

File. Permite abrir un nuevo proyecto, abrir uno existente, grabar, etc. Destaca la opción *Import*, la cual permite abrir un archivo en formato ARFF para trabajar con él. Además se tiene la opción *Export*, que permite, dado un cluster, crear un archivo *.ARFF* con sus datos.

Edit. Ofrece las opciones típicas de cortar, pegar, etc.

Build. Esta opción engloba todo lo referente al proceso de clustering:

[1] http://www.dplot.com.

Separate Test Instances Para el caso en el que se vaya a realizar una prueba de test, es necesario separar previamente una serie de instancias, que no participan en el proceso de modelización y que más tarde se utilizan para probar el modelo obtenido.

Create Partitions. Es posible, mediante la elección de esta opción, realizar el particionado de un grupo de proyectos, utilizando los valores de un atributo nominal. Es decir, si por ejemplo se tiene el atributo *Used Methodology* (con valores "yes" y "no"), sería posible separar los proyectos con valor "yes" y los que tienen valor "no", dando lugar a dos particiones. Esta opción permite incluir, en la realización de un experimento, la experiencia de expertos.

Create Clusters. Inicia el proceso de clusterización.

Delete Clusters. Estando situado sobre un nodo del árbol de clusters, deshace el proceso de segmentación realizado anteriormente sobre dicho nodo.

Data Analysis. Permite realizar el proceso de análisis de datos: curva de regresión y valores MMRE y PRED(l). Es necesario estar situado sobre el nodo del que se desee realizar el análisis.

Cluster Link. Es necesario realizar este paso, previo a la realización del siguiente (*Full Test*), para poder obtener una estimación previa de la variable dependiente, que permita en el proceso de test asignar las instancias al cluster correcto. Para ello hay que situarse encima de todos y cada uno de los clusters que han sido clusterizados a su vez y seleccionar esta opción.

Full Test. Una vez realizado el *Cluster Link*, de forma apropiada, y habiendo seleccionado el nodo raíz, esta opción aparece disponible para su utilización. Ofrece los datos completos del test realizado sobre el modelo obtenido, utilizando los proyectos de test separados al principio.

Graph. Permite dibujar los proyectos de los clusters obtenidos, mediante la opción *Plot*. Con *Redraw* se puede redibujar el gráfico, tras un cambio de escala.

View. Dentro de esta opción de menú tenemos otras que permiten ocultar/visualizar

las diferentes barras del programa.

Help. Posibilita el acceso a la ayuda de la herramienta.

A.2. Sesión de trabajo con RCT

En esta sección se va a describir lo que podría ser una sesión de trabajo normal con RCT: abrir un archivo de proyectos, separar las instancias de entrenamiento y de test (si se desea llevar a cabo el mismo), realizar el proceso de clustering sobre las instancias de entrenamiento, el posterior análisis de cada uno de los clusters obtenidos así como el análisis de los resultados obtenidos sobre las instancias de test, y finalmente la representación gráfica de los clusters dados.

Para comenzar a trabajar se puede abrir un archivo ARFF, mediante la importación del mismo (*File* → *Import*). Una vez que está importado ya se podría grabar el proyecto (archivo con extensión *rclt*). Otra posibilidad consistiría en abrir un proyecto existente (un archivo *rclt* cualquiera, con *File* → *Open*).

Ya sea de una forma u otra, cuando se dispone de los datos, éstos se muestran en *Data Bar*, y en la ventana principal aparece el árbol con un único nodo que representa la base de datos de proyectos, tal y como refleja la figura A.1.

Si se desea realizar un análisis final sobre la capacidad predictiva del modelo global obtenido, es necesario en este momento separar un conjunto de proyectos que se utilizarán como instancias de test para realizar dicho análisis. Es posible decidir el número de estos proyectos, los cuales se pueden elegir de forma aleatoria o no de entre todos los existentes. Al realizar esta separación,con la opción *Build* → *Separate Test Instances*, se obtendrán dos conjuntos de proyectos, proyectos de train y proyectos de test (los cuales solamente se tendrán en cuenta en el test final). La figura A.2 muestra esta situación.

A continuación se procedería a realizar el proceso de segmentación sobre el conjunto de datos de entrenamiento. La opción correspondiente es *Build* → *Create Cluster*, estando situados sobre el nodo correspondiente a dichos proyectos. Una vez elegida esta

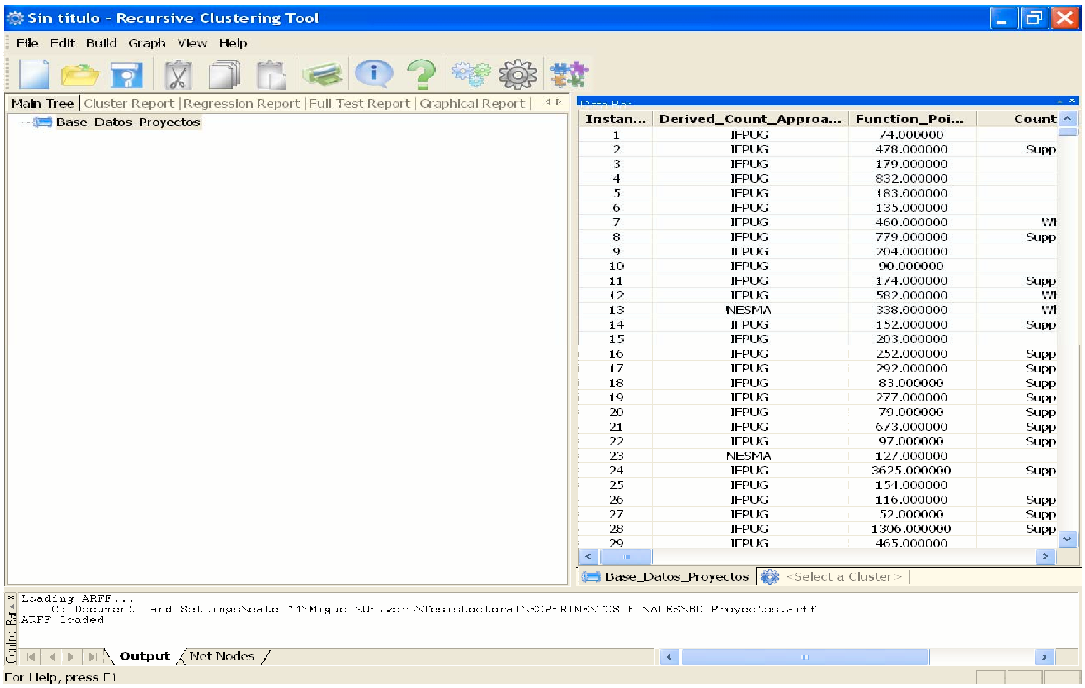

Figura A.1: Pantalla inicial con la base de datos de proyectos cargada

opción, aparece una ventana (figura A.3) que permite seleccionar los atributos por los que realizar el clustering, así como otros parámetros que se verán a continuación.

La ventana de parámetros aparece divida en dos partes: parámetros principales y parámetros adicionales. Dentro de los parámetros principales hay que seguir tres pasos para su selección:

1. Seleccionar atributo o atributos

2. Seleccionar la distribución de probabilidad del atributo o atributos anteriores. Si se trata de un solo atributo se puede elegir distribución normal, log-normal o t-student, sin embargo sólo se podrá elegir distribución normal n-dimensional o t-student si se trata de varios atributos dependientes. Esto es para atributos numéricos, para el caso de atributos nominales no es aplicable, sólo se aplicaría una distribución nominal, específica para ellos

3. Añadir a la función de densidad de probabilidad (*mixture model*), la selección

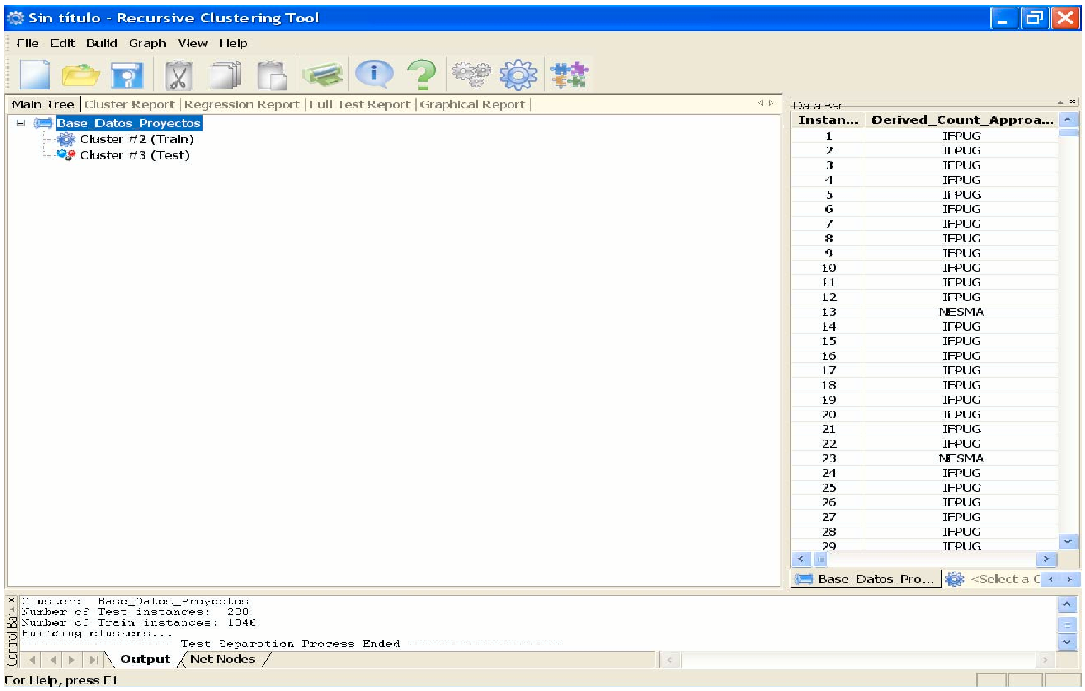

Figura A.2: Pantalla inicial tras la separación de proyectos de training y de test

anterior

Estos pasos se pueden repetir de forma arbitraria, según el número de distribuciones que formen la función de densidad de probabilidad a ajustar por el algoritmo EM. Una vez que se tiene modelada esta función tal y como se desea, se pasa a introducir los parámetros adicionales que se deseen:

Number of Splits for Cross Validation: Número de divisiones que se realizan en el proceso de v-fold cross validation para obtener el valor del número de clusters óptimo. El máximo de los valores ofrecidos como posibilidad, sería la mejor opción, aunque la más costosa computacionalmente. Si se utiliza esta opción no es posible utilizar el criterio *MDL*.

Maximum Number of Clusters: Se selecciona una cota superior. Para los datos que se manejan en el presente trabajo, un número de 15 clusters sería apropiado.

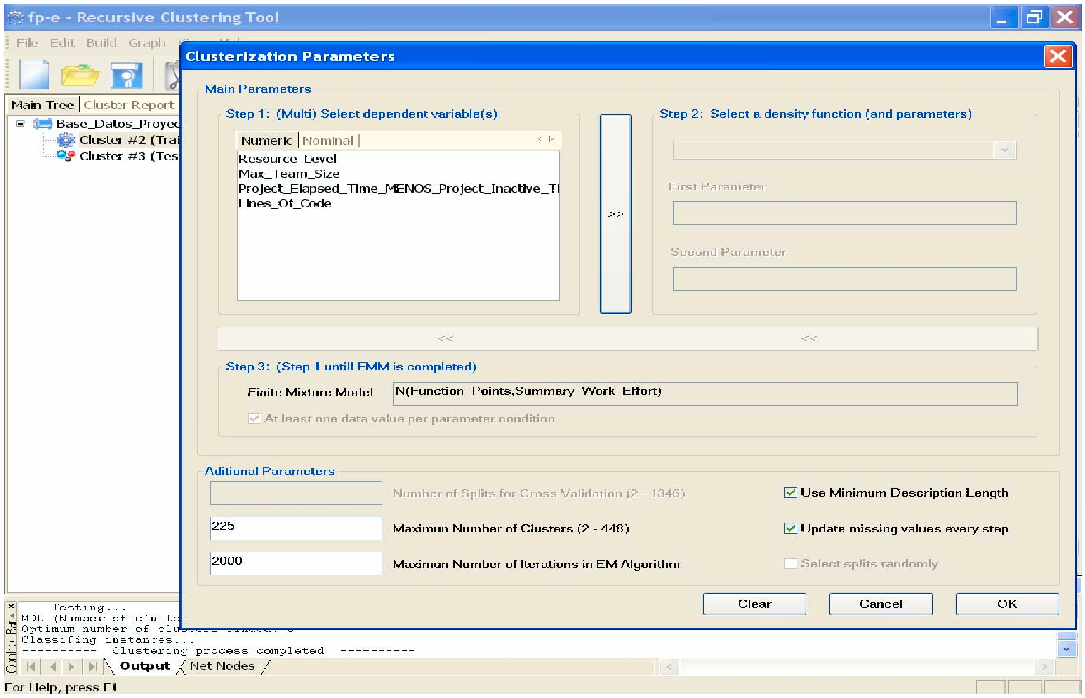

Figura A.3: Ventana de selección de los parámetros de la segmentación

Maximum Number of Iterations in EM algorithm El algoritmo EM tiene un criterio de parada, que es posible que no se cumpla, en cuyo caso como máximo, se ejecutará el número de iteraciones establecido por este parámetro. Garantiza la finalización del algoritmo.

Use Minimum Description Length Por defecto *sí*, indica que este criterio (*MDL*) será el que se utilize para obtener el valor del número de clusters óptimo. Si está seleccionado no es posible utilizar cross-validation.

Update missing values every step El algoritmo EM permite obtener valores para atributos con valores nulos. Esta opción, si se activa, permite ir obtenido valores para dichos casos.

Select splits randomly Esta opción es muy importante ya que según esté activa o no, los clusters obtenidos pueden ser muy diferentes. Si se activa, se seleccionan de forma aleatoria los proyectos que servirán de semilla para el algoritmo EM. Si

no está activa, se seleccionan los primeros de la base de datos, tal y como estén situados inicialmente. Si los proyectos en la base de datos no tienen ningún orden previo, da igual que se seleccionen de forma aleatoria o no los proyectos que van a formar parte de cada uno de los conjuntos. Sin embargo, si existe algún tipo de relación entre los proyectos, según su situación, esta opción debería de estar activada.

Una vez seleccionados todos los parámetros, si la selección no es la deseada, es posible limpiarla y comenzar otra vez, a través del botón *Clear*. Si todo está bien, pulsar *OK*. Una vez aceptada la selección, se procede al proceso de clusterización, apareciendo en *Control Bar* todos los pasos seguidos. Finalmente en la ventana principal aparece bajo la pestaña *Main Tree* el árbol de clusters obtenido a partir del nodo raíz de proyectos de training (figura A.4), y bajo la pestaña *Cluster Report* aparece el informe sobre el proceso realizado (figura A.5).

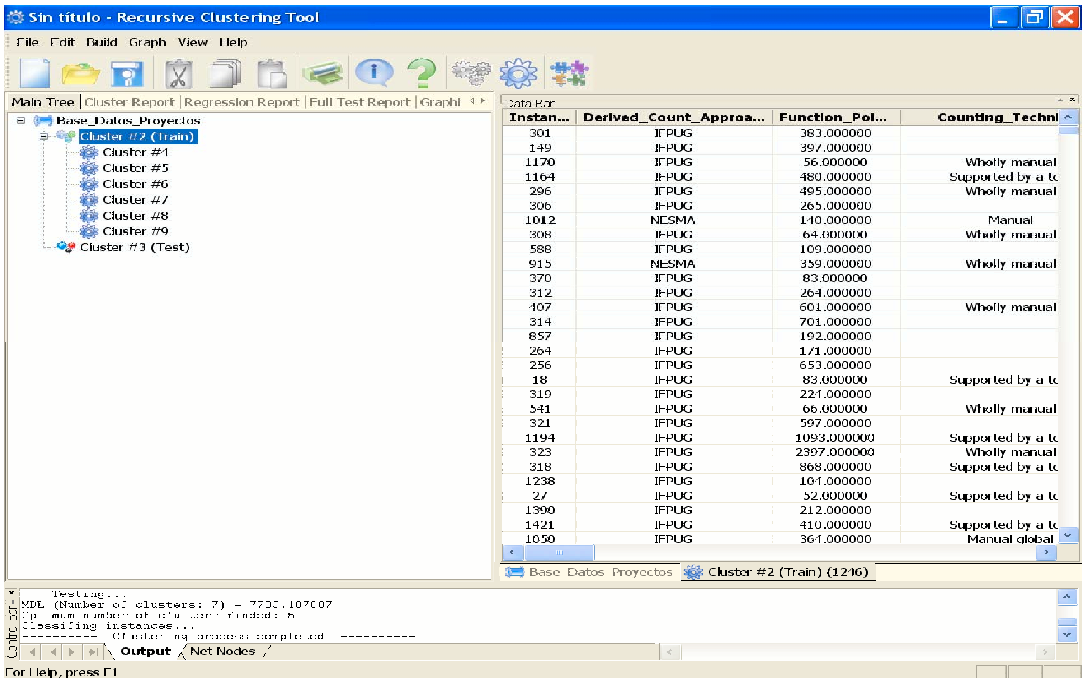

Figura A.4: Árbol de clusters obtenidos

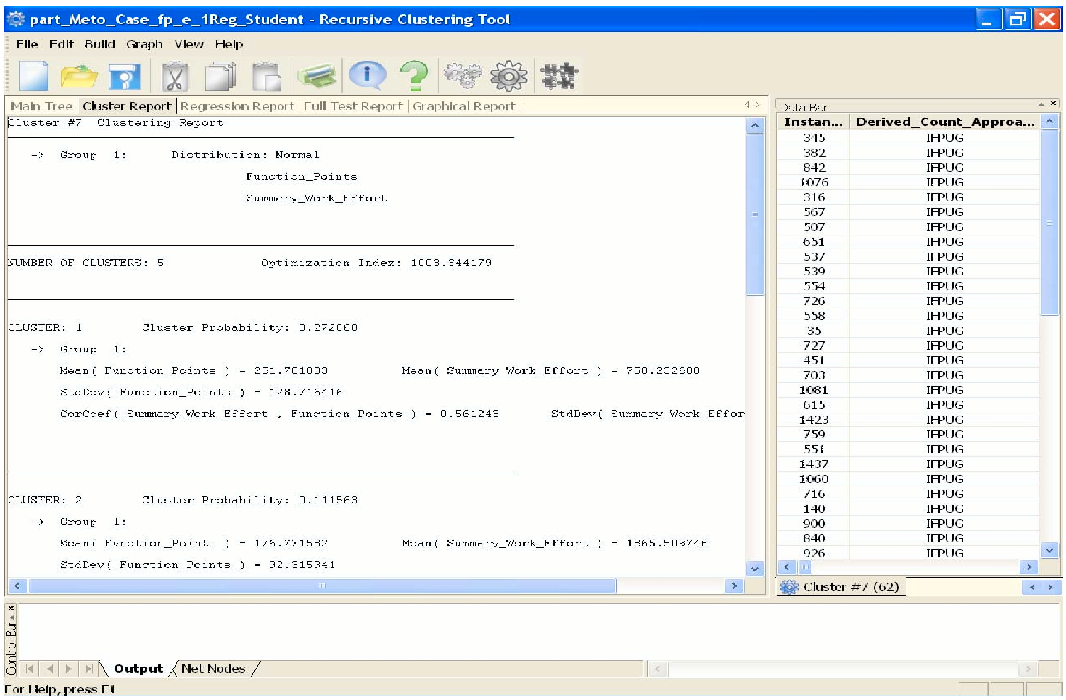

Figura A.5: Informe del proceso de clustering

Una vez clusterizados los proyectos de software, se pueden representar gráficamente o analizar. El proceso de análisis consiste en seleccionar un conjunto de proyectos (ya sea los pertenecientes a un cluster o todos los proyectos de la base de datos), y proceder a la obtención de su curva de regresión y a la validación de la misma. Para ello se selecciona, por ejemplo, un nodo cualquiera y se selecciona la opción *Build* → *Data Analysis*, tras lo cual aparece nuevamente una ventana (figura A.6) que permite seleccionar los parámetros del análisis.

Esta ventana se divide en dos partes, en la primera (*Fitting Parameters*) se selecciona el atributo dependiente, el atributo independiente y la forma de la curva de regresión:

1. Linear (least-absolute-deviation)
2. Linear (least-squares)
3. Linear log-log Plot (least-absolute-deviation)
4. Linear log-log Plot (least-squares)

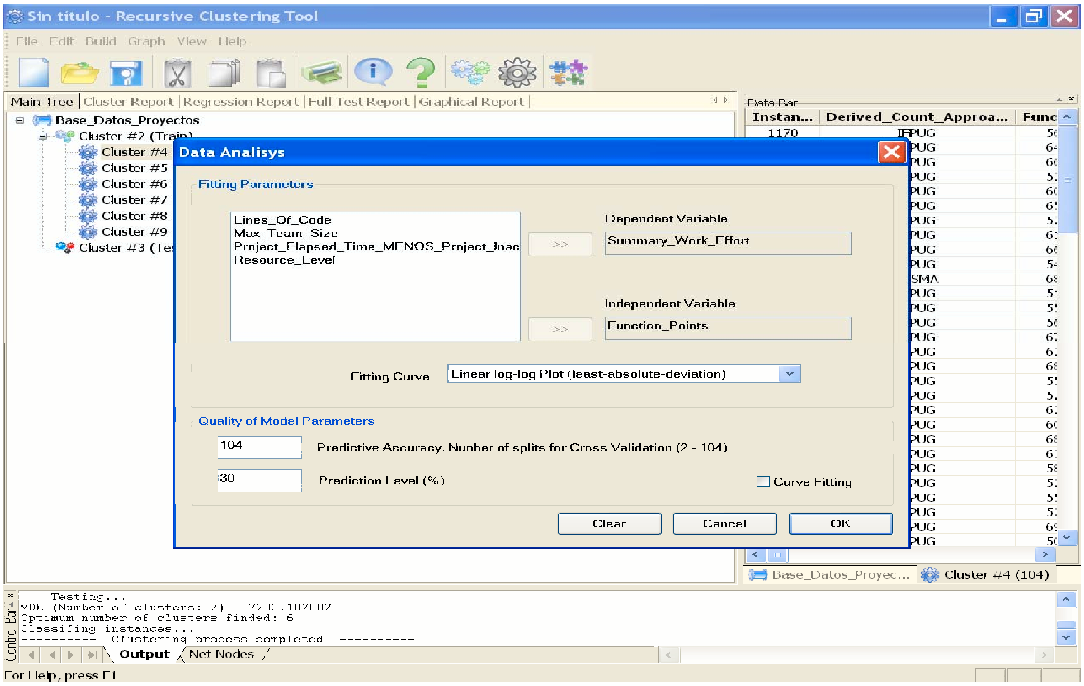

Figura A.6: Selección de los parámetros del análisis de regresión

5. Non-Linear Power (least-squares)

que ya han sido descritas en el capítulo 5.

La segunda parte de la ventana *Quality of Model Parameters*, permite seleccionar el número de particiones que se realizarán con cross-validation. Por otra parte se puede seleccionar el valor que se utilizará para calcular el valor de PRED(l), por ejemplo un valor de 30 para *Prediction Level* calcularía PRED(0,3). Finalmente si todo está correcto se selecciona *OK*, en caso contrario *Clear* y volver a empezar. La casilla de verificación *Curve Fitting*, si se selecciona, permite obtener y probar la curva de regresión que se calcula, con los mismos datos, es decir, no realiza *cross-validation*. Esto mediría la calidad del ajuste de la curva obtenida. El resultado del proceso seguido se puede apreciar en la figura A.7, bajo la pestaña *Regression Report*.

Para realizar el análisis sobre todos los clusters obtenidos, habría que ir seleccionándolos de uno en uno y llevar a cabo las acciones descritas anteriormente, para cada uno de

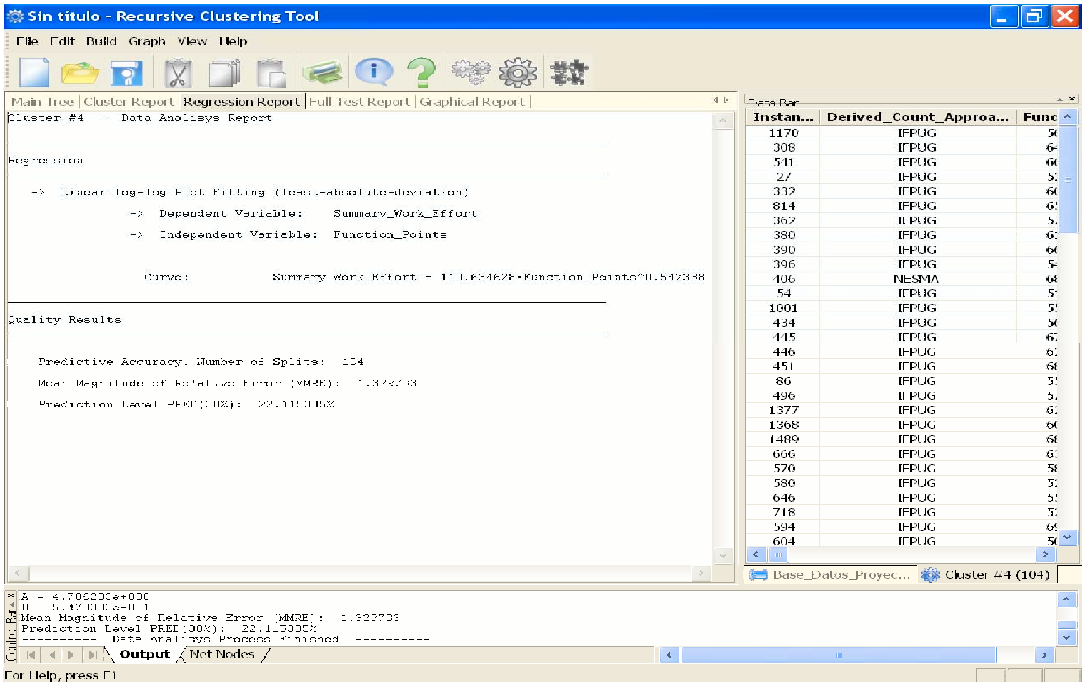

Figura A.7: Resultados del proceso de análisis

ellos.

Una vez que se han obtenido las curvas de regresión de cada uno de los nodos terminales, así como de sus padres, es necesario, siempre que se desee realizar el test final, realizar el proceso de *Cluster Link*, para ello hay que posicionarse sobre cada nodo que se ha clusterizado (no sobre los que se han particionado, si es que los hubiera) y elegir la opción *Build* → *Cluster Link*.

Después de todo esto, ya es posible realizar el test, con los datos que se separaron inicialmente, sobre el modelo obtenido. Para ello es necesario situarse sobre el nodo raíz, el que representa a todos los proyectos (training y test), y seleccionar *Build* → *Full Test*. A continuación, pulsando sobre la pestaña *Full Test Report* aparece toda la información de este test, el cual comprueba la capacidad predictiva real del modelo de estimación segmentado obtenido. Esta información se puede ver en la figura A.8.

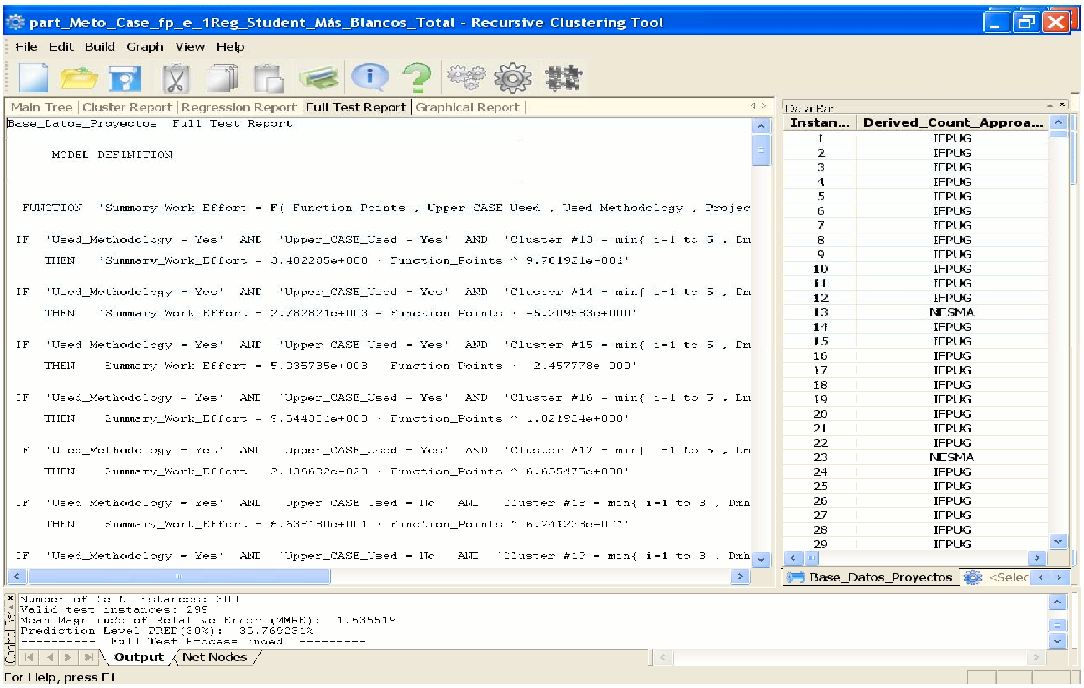

Figura A.8: Resultados del proceso de test

Finalmente, se pueden visualizar de forma gráfica cualquiera de las ramas de clusters obtenidos. Si se desea ver todo el conjunto de clusters, habría que posicionarse sobre el nodo raíz. Si sólo se desean ver los clusters de otro dado, habría que posicionarse sobre éste último, etc., y siempre teniendo en cuenta que se visualizan los clusters que aparecen desplegados en el árbol de clusters bajo la etiqueta *Main Tree*, los no desplegados no aparecerán. La opción que es necesaria seleccionar para la visualización gráfica es *Graph→Plot*, apareciendo un gráfico del estilo de la figura A.9.

Como ya se comentó en la sección A.1, si se desea se puede exportar el gráfico obtenido, para una utilización más cómoda y potente, con DPlot Jr.

Para finalizar la sesión de trabajo, únicamente restaría grabar el proyecto, para poder utilizarlo posteriormente, y salir.

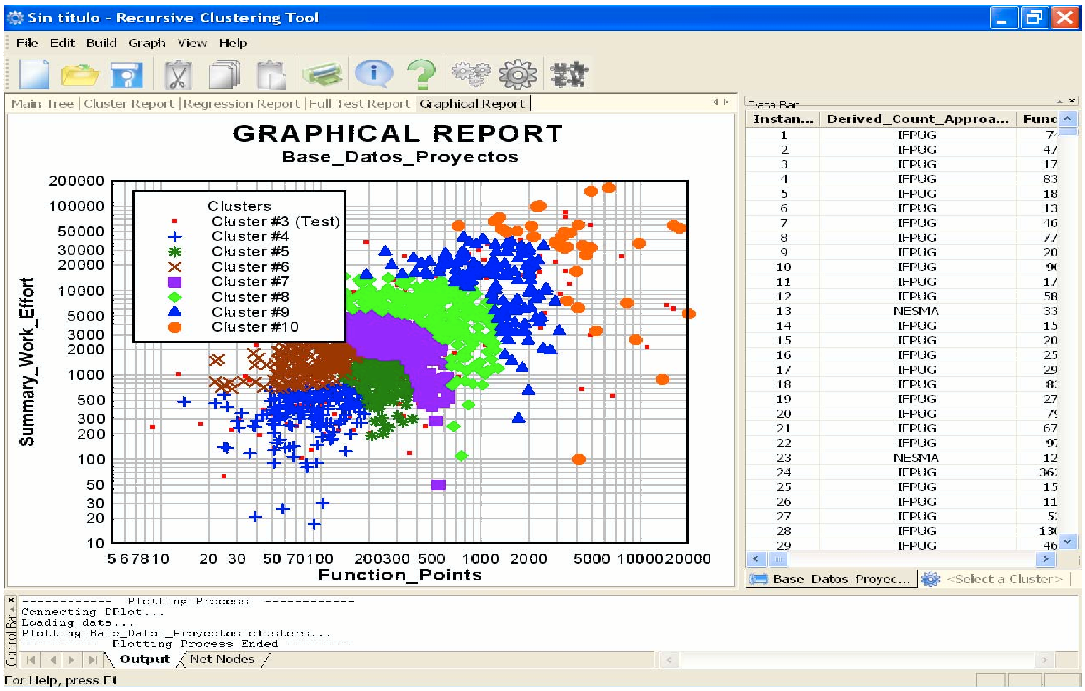

Figura A.9: Gráfico con los clusters obtenidos

A.3. Generalización del proceso

Las posibilidades de RCT son muchas, ya que permite realizar multitud de diferentes combinaciones, tanto con los parámetros de clusterización (sección A.3.1), como con los correspondientes de regresión (sección A.3.2). Además es posible realizar diferentes pruebas de interés con los datos tal y como se verá en la sección A.3.3.

A.3.1. Parámetros de clusterización

A la hora de elegir los parámetros necesarios para realizar el proceso de segmentación, se pueden realizar diferentes selecciones:

- Lo primero que se puede hacer es, seleccionar diferentes atributos, para obtener los clusters. Se pueden elegir, tanto atributos numéricos como nominales, estando cada grupo por separado. Esto permite clusterizar unas veces con unos atributos y otras con otros.

- Para cada atributo seleccionado se elige la distribución de probabilidad, que se considera más apropiada para él, permitiendo realizar diferentes experimentos con el mismo atributo, pero con diferentes distribuciones.

- Además es posible clusterizar suponiendo que varios atributos son dependientes (siguen una distribución normal n-dimensional, o t-student) o que esos mismos atributos son independientes, o combinaciones de estas posibilidades, lo cual proporciona gran riqueza de experimentación.

- Los parámetros adicionales también ofrecen gran riqueza de posibilidades, ya que cambiando su configuración es posible obtener resultados diferentes. Los dos parámetros, quizás más relevantes, son *Number of Splits*, cuyo aumento puede mejorar los resultados, y *Select splits randomly*, que asegura un buen comportamiento estadístico de los datos.

Realizando combinaciones de los valores de estos parámetros se pueden obtener multitud de resultados diferentes.

A.3.2. Parámetros de la curva de regresión

Por otra parte, a la hora de seleccionar los parámetros para obtener la curva de regresión de cada uno de los clusters, se pueden combinar cuatro aspectos:

- Se puede seleccionar entre diferentes atributos, que formarán parte de la curva de regresión, los cuales no tienen porqué ser los mismos que se utilizaron para realizar la segmentación inicial.

- Se puede elegir de entre las diferentes aproximaciones, vistas en la sección A.2, para obtener la correspondiente curva de regresión.

- Se puede seleccionar el número de splits para realizar la validación cruzada.

- Si lo que se desea es comprobar el ajuste de la curva obtenida con los datos utilizados, también será posible a través del punto de validación *Curve Fitting*.

- Por último, es posible determinar el argumento de PRED(l), con *Prediction Level*, para ser más o menos restrictivo en su cálculo.

Así mismo, combinaciones de estas variaciones pueden dar lugar a otros tantos resultados diferentes.

A.3.3. Misceláneas

Para facilitar el análisis de los clusters obtenidos, se han incluido una serie de funcionalidades que es necesario exponer a continuación:

- Una vez realizada la segmentación, es posible, dado un proyecto de la base de datos de proyectos, ver en qué cluster está situado. Para ello es necesario situarse en el nodo que representa la base de datos de proyectos, y en *Data Bar* hacer doble click sobre el proyecto deseado. En ese momento se coloreará en rojo el nodo correspondiente al cluster que contiene dicho proyecto.

- Los datos que aparecen en *Data Bar* se pueden ordenar de forma ascendente o descendente por un determinado atributo. Para ello, se pulsa con un solo click sobre el nombre del atributo, que aparece como título de la columna. Cada vez que se haga click sobre él se cambiará el orden, de forma ascendente a descendente y viceversa.

- Si se desea averiguar en qué clusters están distribuidos los proyectos con valor *NESMA* para el atributo *Derived Count Approach*, por ejemplo, se procedería de la siguiente forma: habría que ordenar la base de datos por ese atributo, seleccionar todos los proyectos con valor *NESMA*, y a continuación ir seleccionando sobre el árbol de segmentos, cada uno de los clusters y buscar en *Data Bar* los proyectos de cada segmento que aparecen resaltados, los cuales serán los buscados.

Como se ha podido observar, esta herramienta es de gran versatilidad y potencia, permitiendo, con tiempo y paciencia, realizar numerosos experimentos, los cuales servirán de ayuda al proceso de predicción en Ingeniería del Software. No es posible detallar

aquí todos los posibles experimentos a realizar, sino describir sus posibilidades, para poder utilizarlas cuando se desee.

Bibliografía

Abdel-Hamid, T. (1989*a*), 'The dynamics of software project staffing:A system dynamics-based simulation approach', *IEEE Transactions on Software Engineering* .

Abdel-Hamid, T. (1989*b*), Lessons learned from modeling the dynamics of software development, *in* 'Comunications of the ACM', ACM.

Abdel-Hamid, T. (1993), 'Adapting, correcting, and perfecting software estimates: a maintenance metaphor', *IEEE Computer* .

Abdel-Hamid, T. K. & Madnick, S. E. (1986), 'Impact of schedule estimation on software project behavior', *IEEE Software* **3**(4), 70–75.

Abdel-Hamid, T. & Madnick, S. (1991), *Software Project Dynamics*, Prentice Hall.

Abdel-Hamid, T. & Madnick, S. (1993), Modeling the dynamics of software reuse: an integrating system dynamics perspective, *in* 'Sixth Annual Workshop on Reuse', Owego, NY.

Abran, A. (2003), Software estimation: Black box or white box, *in* 'Workshop ADIS'.

Aggarwal, C., Procopiuc, C., Wolf, J., Yu, P. & Park, J. (1999), Fast algorithms for projected clustering, *in* 'Proceedings of the ACM SIGMOD Conference', Philadelphia, PA, pp. 61–72.

Aggarwal, C. & Yu, P. (2000), 'Finding generalized projected clusters in high dimensional spaces', *Sigmod Record* **29**(2), 70–92.

Agrawal, R., Gehrke, J., Gunopulos, D. & Raghavan, P. (1998), Automatic subspace clustering of high dmensional data for data mining applications, *in* 'Proceedings of the ACM SIGMOD Conference', Seattle, WA, pp. 94–105.

Aguilar-Ruiz, J. S., Ramos, I., Riquelme, J. C. & Toro, M. (2001), 'An evolutionary approach to estimating software development projects', *Information and Software Technology* **43**, 875–882.

Albrecht, A. & Gaffney, J. (1983), 'Software function, source lines of code, and development effort prediction: a software science validation', *IEEE Transactions on Software Engeneering* **9**(6), 639–648.

Ankerst, M., Breunig, M., Kriegel, H. & Sander, J. (1999), OPTICS: Ordering points to identify clustering structure, *in* 'Proceedings of the ACM SIGMOD Conference', Philadelphia, PA, pp. 49–60.

Arabie, P. & Hubert, L. J. (1996), *An overview of combinatorial data analysis*, Clustering and Classification, World Scientific Publishing Co., New York.

Aron, J. (1969), Estimating Resources for Large Systems, *in* J. Buxton & B. Randell, eds, 'NATO Conference Report on Software Engineering Techniques', Roma, (Italia).

Baik, J., Chulani, S. & Boehm, B. (1998), The Effects of CASE Tools on Software Development Effort, *in* 'Proceedings of 13th International Forum on COCOMO and 11th Focused Workshop on COCOMO II Extensions', University of Southern California.

Bailey, J. W. & Basili, V. R. (1981), A meta-model for software development resource expenditures, *in* '5th International Conference on Software Engineering', Dept. of Computer Sci. Univ. of Maryland College Park MD USA, New York.

Banker, R., Kauffman, R. & Kumar, R. (1994), 'An empirical test of object-based output measurement metrics in a computer aided software engineering environment', *Journal of Management Information System* .

Barbara, D. & Chen, P. (2000), Using the fractal dimension to cluster datasets, *in* 'Proceedings of the 6th ACM SIGKDD', Boston, MA, pp. 260–264.

Bäck, T., Hammel, U. & Schwefel, H. (1997), 'Evolutionary computation: Comments on the history and current state', *IEEE Trans. on Evolutionary Computation* **1**(1), 3–17.

Ben-Dor, A. & Yakhini, Z. (1999), Clustering gene expression patterns, *in* 'Proceedings of the 3rd Annual International Conference on Computational Molecular Biology (RECOMB 99)', Lyon, France.

Berkhin, P. (2002), Survey of clustering data mining techniques, Technical report, Accrue Software, Inc.

Berkhin, P. & Becher, J. (2002), Learning Simple Relations. Theory and Applications, *in* 'Proceedings of the 2nd SIAM ICDM', Arlington, VA, pp. 420–436.

Black, R., Curnow, R., Katz, R. & Gray, M. (1977), BCS Software Production Data, Final Technical Report RADC-TR-77-116, Boeing Computer Services Inc.

Boehm, B. (1981), *Software Engineering Economics*, Vol. 10, Prentice-Hall.

Boehm, B., Abts, C. & Chulani, S. (2000), 'Software development cost estimation approaches - a survey', *Annals of Software Engineering* **10**, 177–205.

Boehm, B., Clark, B., Horowitz, E., Madachy, R., Selby, R. & Westland, C. (1995), *Cost Model for Future Software Life Cycle Processes: COCOMO 2.0*, Vol. 1, AG Science Publishers, Amsterdam (Holanda). En Annals of Softwaer Engineering Special Volume on Software Process and product Measurement.

Boehm, B., Horowitz, E., Madachy, R., Reifer, D., Clark, B., Steece, B., Brown, W., Chulani, S. & Abts, C. (2000), *Software Cost Estimation with Cocomo II*, Prentice Hall.

Boehm, B. W. (1984), 'Software engineering economics', *IEEE Transactions on Software Engineering* **10**(1), 4–21.

Boley, D. L. (1998), 'Principal direction divisive partitioning', *Data Mining and Knowledge Discovery* **2**(4), 325–344.

Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1993), *Classification and Regression Trees*, Chapman and Hall, New York.

Briand, L. C., Basili, V. R. & Thomas, W. M. (1992), 'A pattern recognition approach for software engineering data analysis', *IEEE Transactions on Software Engineering* **18**(11), 931–942.

Briand, L. C., El Emam, K., Maxwell, K., Surmann, D. & Wieczorek, I. (1999), An assessment and comparison of common cost software project estimation methods, *in* 'Proc. International Conference on Software Engineering, ICSE 99', pp. 313–322.

Briand, L. C., Emam, K. & Bomarius, F. (1998), Cobra: a hybrid method for software cost estimation, benchmarking, and risk assessment, *in* 'International Conference on Software Engineering', IEEE Comput. Soc., Kyoto, Japan, pp. xiv+521.

Briand, L., Langley, T. & Wieczorek, I. (2000), Using the European Space Agency data set: A replicated assessment and comparison of common software cost modeling, *in* 'Proceedings of the 22th International Conference on Software Engineering', Limerik, Ireland, pp. 377–386.

Buck, F. (1971), *A Cost-by-Function Model for Avionics Computer System*, NADC-SD-7088 Vol. 1 edn, Naval Air Development Center.

Cadez, I., Smyth, P. & Mannila, H. (2001), Probabilistic modeling of transactional data with applications to profiling, visualization, and prediction, *in* 'Proceedings of the 7th ACM SIGKDD', San Francisco, California, pp. 37–46.

Cheeseman, P. & Stutz, J. (1996), *Bayesian Classification (AutoClass): Theory and Results*, AAAI Press/MIT Press.

Cheng, C., Fu, A. & Zhang, Y. (1999), Entropy-based subspace clustering for mining numerical data, *in* 'Proceedings of the 5th ACM SIGKDD', San Diego, CA, pp. 84–93.

Chulani, S. (1998), Incorporating Bayesian Analysis to improve the Accuracy of COCOMO II and its Quality Model Extension, Ph.D. Qualifying Exam Report, University of Southern California.

Chulani, S., Boehm, B. & Steece, B. (1998), Calibrating Software Cost Models using Bayesian Analysis, Technical Report, USC-CSE-98-508. To appear in IEEE Transactions on Software Engineering, Special Issue on Empirical Methods in Software Engineering.

Conte, S. D., Dunsmore, H. E. & Shen, V. Y. (1986), *Software Engineering Metrics and Models*, Benjamin-Cumming Co., Inc. Menlo Park CA.

Cooley, R., Mobasher, B. & Srivastava, J. (1999), 'Data preparation for mining world wide web browsing', *Journal of Knowledge Informaton Systems* **1**(1), 5–32.

Cowderoy, A. J. & Jenkins, J. O. (1988), 'Cost estimation by analogy as a good management practice', *Software Engineering* (88).

Crespo, J., Sicilia, M. & Cuadrado-Gallego, J. (2003), On fuzzy regression in software cost estimation models, *in* 'International symposium on empirical software engineering (ISESE) 2004 conference proceedings. IEEE Computer Society', Frascati, Roma, Italy.

Crespo, J., Sicilia, M. & Cuadrado-Gallego, J. (2004), Auditing parametric software cost estimation: issues related to introducing fuzziness, *in* '25th Annual conference of the Iternational Society of Parametric Analysts (ISPA) and the 15th Annual conference of the Society of Cost Estimating and Analysis (SCEA) conference proceedings', Frascati, Roma, Italy.

Cuadrado, J., Rodríguez, D. & Sicilia, M. (2004), 'Modelos segmentados de estimación del esfuerzo de desarrollo del software: un caso de estudio con la base de datos ISBSG', *Revista de Procesos y Métricas de las Tecnologías de la Información (RPM)* **1**(2), 25–30. ISSN: 1698-2029.

Cuadrado, J., Sicilia, M., Garre, M. & Rodríguez, D. (2006), 'An empirical study of process-related attributes in segmented software cost-estimation relationships', *Journal of Systems and Software* **79**(3), 351–361. ISSN: 0164-1212. Incluido en el Journal of Citation Reports (JCR) 2004, Computer Science, Software Engineering. Factor de Impacto: 0.716. Ranking 36 de 76. Editorial: Elsevier.DOI information: 10.1016/j.jss.2005.04.040.

Cutting, D., Karger, D., Pedersen, J. & Tukey, J. (1992), Scatter/gather: a cluster-based approach to browsing large document collection., *in* 'Proceedings of the 15th ACM SIGIR Conference', Copenhage, Denmark, pp. 318–329.

Dadoda, G., Cartwright, M., Chen, L. & Shepperd, M. (2000), Experiences using case-based reasoning to predict software project effort, *in* 'Proceedings of EASE', Keele, UK, pp. 23–28.

Daniel, C. & Wood, F. C. (1980), *Fitting Equations To Data: Computer Analysis of Multifactor Data*, John Wiley & Sons, New York, NY.

DeMarco, T. (1982), *Controlling Software Projects*, Yourdan Press.

Dempster, A., Laird, N. & Rubin, D. (1977), 'Maximun likelihood from incomplete data via the EM algorithm', *Journal of the Royal Statistical Socety* **39**(1), 1–38.

Desharnais, J. M. (1989), Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction, Master's thesis, University of Montreal.

Dhillon, I. an Guan, Y. & Kogan, J. (2002), Refining clusters in high dimensional data, *in* '2nd SIAM ICDM, Workshop on clustering high dimensional data', Arlinton, VA.

Dhillon, I., Fan, J. & Guan, Y. (2001), *Efficient clustering of very large document collections*, Data Mining for Scientific and Engineering Applictions, Kluwer Academic Publishers.

Dolado, J. & Fernández, L. (1998), Genetic programming, neural networks and linear regression in software project estimation, *in* 'INSPIRE III, Process Improvement thorough Training and Education', The British Computer Society, pp. 155–171.

Dolado, J. J. (1998), 'Limits to the methods in software cost estimation'.

Dolado, J. J. (2001), 'On the problem of the software cost function', *Information and Software Technology* **43**, 61–72.

Dolado, J. J., Fernández, L., Otero, M. C. & Urkola, L. (1998), 'Software effort estimation: The elusive goal in project management'.

Dreger, J. (1989), *Function Point Analysis*, Englewood Cliffs, NJ: Prentice Hall.

Duda, R. & Hart, P. (1973), *Pattern Classification and Scene Analysis*, John Wiley & Sons.

Ertoz, L., Steinbach, M. & Kumar, V. (2002), Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data, Technical Report.

Ester, M., Kriegel, H. & Xu, X. (1996), A database interface for clustering in large spatial databases, *in* 'Proceedings of the 1st ACM SIGKDD', Montreal, Canada, pp. 94–99.

Fairley, R. E. (1992), Recent advances in software estimation techniques, *in* 'International Conference on Software Engineering, Software Eng. Manage', Associates Woodland Park CO USA. ACM, New York NY USA.

Farr, L. & Zagorski, H. J. (1965), Quantitative analysis of programming cost factors: A progress report, *in* 'ICC Symposium Proc. Economics of Automatic Data Processing', North-Holland, Amsterdam.

Finnie, G. R., Wittig, G. E. & Desharnais, J. (1997), 'A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models', *Journal of Systems and Software* **39**(3), 281–289.

Fisher, D. (1987), 'Knowledge acquisition via incremental conceptual clustering', *Machine Learning* **2**, 139–172.

Forgy, E. (1965), 'Cluster analysis of multivariate data: Efficiency versus interpretability of classification', *Biometrics* **21**, 768–780.

Forrester, J. (1961), *Industrial Dynamics*, MIT Press, Cambridge, MA.

Fraley, C. & Raftery, A. (1999), MCLUST: Software for model-based cluster and discriminant analysis, Tech report 342, Dept. Statistics, Univ. of Washington.

Frederic, B. C. (1974), *A Provisional Model for Estimating Computer Program Development Costs*, TECOLOTE Research Inc. En TM-7/REV-1.

Ganti, V., Gehrke, J. & Ramakrishnan, R. (1999), CACTUS-Clustering Categorical Data Using Summaries, *in* 'Proceedings of the 5th ACM SIGKDD', San Diego, CA, pp. 73–83.

Garre, M. & Charro, M. (2005), 'Estimación del esfuerzo de un proyecto software utilizando el criterio MDL-EM y componentes normales N-Dimensionales. Aplicación a un caso práctico', *Revista de Procesos y Métricas de las Tecnologías de la Información (RPM)* **2**(1), 13–24. ISSN: 1698-2029.

Garre, M., Cuadrado, J. & Sicilia, M. (2004), Recursive segmentation of software projects for the estimation of development effort, *in* 'Proceedings of the ADIS 2004 Workshop on Decision Support in Software Engineering, CEUR Workshop proceedings', Vol. 120. Disponible en http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-120.

Garre, M., Cuadrado, J. & Sicilia, M. (2006), 'Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software', *Revista para la Experimentación y Medición en Ingeniería del Software (REMIS)* . En prensa. ISSN: 1886-2918.

Garre, M., Cuadrado, J., Sicilia, M., Charro, M. & Rodríguez, D. (2005), Segmented Parametric Software Estimation Models: Using the EM algorithm with the ISBSG 8 database, *in* 'Information Technology Interfaces'.

Gersho, A. & Gray, R. M. (1992), *Vector Quantizaton and Signal Compression Communications and Informaton Theory*, Kluwer Academic Publishers, Norwell, MA.

Gibson, D., Kleinbeg, J. & Raghavan, P. (1998), Clustering categorical data: An approach based on dynamic systems, *in* 'Proceedings of the 24th International Conference on Very Large Databases', New York, NY, pp. 311–323.

Goil, S., Nagesh, H. & Choudhary, A. (1999), MAFIA: Efficient and scalable subspace clustering for very large data sets, Technical report, Northwestern University CPDC-TR-9906-010.

Govaert, G. (1995), 'Simultaneous clustering of rows and columns', *Control and Cybernetics* **24**, 437–458.

Gray, A. & McDonell, S. (1997), 'A Comparison of Techniques for Developing Predictive Models of Software Metrics', *Information and Software Technology* **39**.

Guha, S., Rastogi, R. & Shim, K. (1998), CURE: An efficent clustering algorithm for large databases, *in* 'Proceedings of the ACM SIGMOD Conference', Seattle, WA, pp. 73–84.

Guha, S., Rastogi, R. & Shim, K. (1999), ROCK: A robust clustering algorithm for categorical attributes, *in* 'Proceedings of the 15th ICDE', Sydney, Australia, pp. 512–521.

Han, J. & Kamber, M. (2001), *Data Mining*, Morgan Kaufmann Publishers.

Hartigan, J. (1975), *Clustering Algorithms*, John Wiley & Sons, New York, NY.

Hartigan, J. & Wong, M. (1979), 'Algorithm AS139: A k-means clustering algorithm', *Applied Statistics* **28**, 100–108.

Hartly, H. (1958), 'Maximum likelihood estimation from incomplete data sets', *Biometrics* **14**, 174–194.

Heemstra, F. J. (1992), 'Software cost estimation', *Information and Software Technology* **34**(10), 627–639.

Heiat, A. (2002), 'Comparison of artificial neural network and regression models for estimating software development effort', *Information and Software Technology* (44), 911–922.

Helmer, O. (1966), *Social Technology*, Basic Books.

Herd, J., Postak, J., Russell, W. & Stewart, K. (1977), Software Cost Estimation Study - Study Results, Final Technical Report, Doty Associates, Inc.

Hinneburg, A. & Keim, D. (1998), An efficient approach to clustering large multimedia databases with noise, *in* 'Proceedings of the 4th ACM SIGKDD', New York, NY, pp. 58–65.

Hinneburg, A. & Keim, D. (1999), Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering, *in* 'Proceedings of the 25th Conference on VLDB', Edinburgh, Scotland, pp. 506–517.

Holgeid, K., Krogstie, J. & I.K., S. D. (2000), 'A study of development and maintenance in norway: assessing the efficiency of information systemas support using funcional maintenance', *Information and Software Technology* (42), 687–700.

Ian, H. & Eibe, F. (2000), *Data Mining. Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers, San Francisco, California.

Idri, A. & Abran, A. (2002), 'Fuzzy case-based reasoning models for software cost estimation'.

Idri, A., Abran, A. & Khoshgoftaar, T. M. (2001), Fuzzy analogy: A new approach for software cost estimation, *in* 'Proceedings of the 11th International Workshop on Software Measurements', Montreal, Canada, pp. 93–101.

Jager, R. (1995), Fuzzy Logic in Control, PhD thesis, Technic University Delft, Holland.

Jain, A. & Dubes, R. (1988), *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, NJ.

Jain, A. K. & Flynn, P. J. (1966), 'Image segmentation using clustering.', *IEEE Press* .

James, T. J. (1977), Software Cost Estimating Methodology, *in* 'IEEE Proceedings of National Aerospace Electronic Conference', pp. 22–28.

Jensen, R. (1983), An improved macrolevel software development resource estimation model, *in* 'Proceedings 5th ISPA Conference', pp. 88–92.

Jones, C. (1996), *Applied software measurement*, McGraw-Hill.

Jorgensen, M. (1995), 'Experience with the accuracy of software maintenance task effort prediction models', *IEEE Transactions on Software Engeneering* **21**(8), 674–681.

Jorgensen, M. (2002), 'A review of studies on expert estimation of software development effort', *The Journal of Systems and Software* **70**, 37–60.

Judge, G., Griffiths, W. & Hill, C. (1993), *Learning and Practicing Econometrics*, Wiley.

Karypis, G., Aggarwal, R., Kumar, V. & Shekhar, S. (1997), Multilevel hypergraph partitioning: application in VLSI domain, *in* 'Proceedings ACM/IEEE Design Automation Conference'.

Karypis, G., Han, E. & Kumar, V. (1999), 'CHAMELEON: A hierarchical clustering algorithm usyng dynamic modeling', *COMPUTER* **32**, 68–75.

Kauffman, R. & Kumar, R. (1993), *Modeling Estimation Expertise in Object Based ICASE Environments*, Ed. Stern School of Business Report, New York University.

Kaufman, L. & Rousseeuw, P. (1990), *Findign Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, New York, NY.

Kemerer, C. (1987), 'An empirical validation of software cost estimation models', *Comm. ACM* **30**(5), 416–429.

Kennedy, H. C., Chinniah, C., Bradbeer, P. & Morss, L. (1997), The construction and evaluation of decision trees: a comparison of evolutionary and concept learning methods, *in* A. J. Corne, D., ed., 'AISB International Workshop', Evolutionary Computing, Springer, LNCS 1305, Manchester, UK.

Kitchenham, B. (1998), 'A procedure for analyzing unbalanced datasets', *IEEE Transactions on Software Engineering* **24**(4), 278–301.

Kitchenham, B. & Kansala, K. (1993), Inter-item correlations among function points, *in* 'Proc. First METRICS, IEEE Computer Society Press', Los Alamitos, California, pp. 11–14.

Kitchenham, B., Pfleeger, S., Hoaglin, D. & Rosenberg, J. (2002), 'Preliminary guidelines for empirical research in software engineering', *IEEE Transactions on Software Engeneering* **28**(8).

Koza, J. R. (1992), *Genetic Programming: On the Programming of Computers by Natural Selection*, MIT Press.

Kustanowitz, A. L. (1977), System life cicle estimation (slice): A new approach to estimating resources for application program development, *in* 'Proccedings of the IEEE COMSAC'.

Leake, D. (1996), CBR in context: The present and the future, *in* M. P. A. Press, ed., 'Case Based Reasoning: Experiences, Lessons and Future Directions', pp. 1–35.

Lee, A., Cheng, C. H. & Balakrishann, J. (1998), 'Software development cost estimation: Integrating neural network with cluster analysis', *Information & Management* (34), 1–9.

Lung, C., Zaman, M. & Nandi, A. (2004), 'Applications of clustering techniques to software partitioning, recovery and restructuring', *Journal of Systems and Software* **73**(2), 227–244.

MacDonell, S. G. & Shepperd, M. J. (2003), 'Combining techniques to optimize effort predictions in software project management', *The Journal of Systems and Software* (66), 91–98.

Madachy, R. (1994), A Software Project Dynamics Model for Process Cost, Schedule and Risk Assessment, Ph.D. dissertation, University of Southern California.

Mair, C., Kadoda, G., Lefley, M., Keith, P., Schofield, C., Shepperd, M. & Webster, S. (2000), 'An investigation of machine learning based prediction systems', *The Journal of Systems and Software* **53**, 23–29.

Marquardt, D. (1963), 'An algorithm for least squares estimation of nonlinear parameters', *Journal of the Society for Industrial and Applied Mathematics* **11**, 431–441.

Massart, D. & Kaufman, L. (1983), *The Interpretation of Analytical Chemical Data by the Use of Cluster Analysis*, John Wiley & Sons, New York.

McKay, B., Elsey, J., Willis, M. & Barton, G. (1996), Evolving input-output models of chemical process systems using genetic programming, *in* 'Proceedings of the IFAC'96', San Francisco.

McKay, B., Willis, M. J. & Barton, G. W. (1997), 'Steady-state modelling of chemical process systems using genetic programming', *Computers and Chemical Engineering* **21**(9), 981–996.

McLachlan, G. & Basford, K. (1988), *Mixture Models: Inference and Applications to Clustering*, Marcel Dekker, New York, NY.

McLachlan, G. & Krishnan, T. (1997), *The EM Algorithm and Extensions*, John Wiley & Sons, New York, NY.

Mitchell, T. (1997), *Machine Learning*, McGraw-Hill, New York, NY.

Mohanty, S. (1981), 'Software cost estimation: present and future', *Software Practice and Experience* **11**(2), 103–121.

Mukhopadhyay, T., Vicinanza, S. S. & Prietula, M. J. (1992), 'Examining the feasibility of a case-based reasoning model for software effort estimation', *Management Information Systems Quarterly* **16**(2), 155–171.

Munakata, T. & Jani, Y. (1994), Fuzzy systems: an overview, *in* 'Communications of the ACM', Vol. 37, pp. 69–76.

Myrtveit, I. & Stensrud, E. (1999), 'A controlled experiment to assess the benefits of estimating with analogy and regression models', *IEEE Transactions on Software Engineering* **25**(4), 510–525.

Nelson, E. (1967), *Management Handbook for the estimation of computer programming costs*, Systems Development Corporation.

NESMA FPA. Counting Practices Manual (CPM 2.0) (1996).

Niessink, F. & Van Vliet, H. (1997), Predicting maintenance effort with function points, *in* 'Proceedings of the International Conference on Software Maintenance', IEEE Computer Society, Bari, Italy.

Norden, P. (1963), *Useful Tool for Project Management*, Wiley.

Oligny, S., Bourque, P., Abran, A. & Fournier, B. (2000), Exploring the relation between effort and duration in software engineering projects, *in* 'Proceedings of the World Computer Congress', Beijing, China, pp. 175–178.

Oyanagi, S., Kubota, K. & Nakase, A. (2001), Application of matrix clustering to web log analysis and access prediction, *in* '7th ACM SIGKDD, WEBKDD Workshop', San Francisco, CA.

Parr, F. (1980), 'An alternative to the Rayleigh curve model for software development effort', *IEEE Transactions on Software Engineering* **6**(3), 291–296.

Putnam, L. H. (1978), 'A general empirical solution to the macro software sizing and estimating problem', *IEEE Transactions on Software Engineering* **4**(4), 345–361.

Putnam, L. & Myers, W. (1992), *Measures for Excellence*, Yourdon Press Computing Series.

Reifer, D., Boehm, B. & Chulani, S. (1999), 'The Rosetta Stone. Making COCOMO 81 Estimates Work with COCOMO II', *CrossTalk* **12**(2), 11–15.

Rissanen, J. (1983), 'A universal prior for integers and estimation by minimum description length', *Annals of Statistics* **2**(11), 417–431.

Rook, P. (1990), *Software reliability handbook*, Elsevier Applied Science, London.

Rubin, H. A. (1983), Macro-estimation of software development parameters: the estimacs system, *in* S. S. M. USA, ed., 'SOFTAIR. A Conference on Software Development Tools Techniques and Alternatives.', IEEE Comput. Soc. Press, Dept. of Computer Sci., Hunter Coll. City Univ. of New York NY USA, pp. xi+395. Proceedings.

Sander, J., Ester, M., Kriegel, H. & Xu, X. (1998), 'Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications', *Data Mining and Knowledge Discovery* **2**(2), 169–194.

Schikuta, E. (1996), Grid-clustering: a fast hierarchical clustering method for very large data sets, *in* 'Proceedings 13th International Conference on Pattern Recognition', Vol. 2, pp. 101–105.

Schikuta, E. & Erhart, M. (1997), The BANG-clustering system: grid-based data analysis, *in* 'Proceeding of Advances in Intelligent Data Analysis, Reasoning about Data, 2nd International Symposium', pp. 513–524.

Scott, D. W. (1992), *Multivariate Density Estimation*, Wiley, New York, NY.

SELECT Software Tools, I. (1998), *Estimation for Component-based Development Using SELECT Estimator*, SELECT Software Tools.

Shan, Y., McKay, R., Lokan, C. & Essam, D. (2002), 'Software project effort estimation using genetic programming'.

Sheikholeslami, G., Chatterjee, S. & Zhang, A. (1998), WaveCluster: A multiresolution clustering approach for very large spatial databases, *in* 'Proceedings of the 24th Conference on VLDB', New York, NY, pp. 428–439.

Shepperd, M. J., C., S. & Kitchenham, B. A. (1996), Effort estimation using analogy, *in* '18th Intl. Conf. on Softw. Eng.', IEEE Computer Press., Berlin.

Shepperd, M. & Kadoda, G. (2001), 'Comparing software prediction techniques using simulation', *IEEE Transactions on Software Engineering* **27**(11).

Shepperd, M. & Schofield, C. (1997), 'Estimating software project effort using analogies', *IEEE Transactions on Software Engineering* **23**(11), 736–743.

Shukla, K. K. (2000), 'Neuro-genetic prediction of software development effort', *Information and Software Technology* (42), 701–713.

Sicilia, M., Cuadrado, J., Crespo, J. & García-Barriocanal, E. (2005), 'Software cost estimation with fuzzy inputs: fuzzy modeling and aggregation of cost drivers', *Kybernetika* (41), 249–264.

Srinivasan, K. & Fisher, D. (1995), 'Machine learning approaches to estimating software development effort', *IEEE Transactions on Software Engineering* **21**(2), 126–137.

Steinbach, M., Karypis, G. & Kumar, V. (2000), A comparison of document clustering techniques, *in* '6th ACM SIGKDD, World Text Mining Conference', Boston, MA.

Stensrud, E., Foss, T., Kitchenham, B. & Myrtveit, I. (2002), An empirical validation of the relationship between the magnitude of relative error and project size, *in* 'Proceedings of the Eighth IEEE Symposium on Software Metrics', Ottawa, Canada, pp. 3–12.

Taback, R. & Ditimore, J. (1974), *Estimation Computer Requeriments and Software Development Costs*, General Research Corporation. En RM-1873.

Tausworthe, R. C. (1981), *Deep Space Network Estimation Model*, report 817 edn, NASA's Jet Propulsion Laboratory.

Thebaut, S. M. (1983), The saturation effect in large-scale software development: its impact and control, Department of Computer Science, Purdue University.

Tung, A., Lakshmanan, L. & Han, J. (2001), Constraint-based clustering in large databases, *in* 'Proceedings of the 8th ICDT', London, UK.

USC-CSE (1997), *COCOMO II Model Definition Manual*, Center for Software Engineering, Computer Science Department, University of Southern California, Los Angeles (USA).

Vicinanza, S. S., Prietula, M. J. & Mukhopadhyay, T. (1990), Case-based reasoning in effort estimation, *in* 'Proc. 11th Intl. Conf. Information Systemas'.

Walkerden, F. & Jeffery, D. R. (1997), 'Software cost estimation: A review of models, process, and practice', *Advances in Computers* **44**, 59–125.

Walston, C. E. & Felix, D. P. (1977), 'A method of programming measurement and estimation', *IBM Systems Journal* **16**(1), 54–73.

Wang, W., Yang, J. & Muntz, R. (1997), STING: a statistical information grid approach to spatial data mining, *in* 'Proceedings of the 23rd Conference on VLDB', Athens, Greece, pp. 186–195.

Weisberg, S. (1985), *Applied Linear Regression*, 2nd edn, John Wiley and Sons, New York, NY.

Wieczorek, I. & Briand, L. (2001), Resource estimation in software engineering, Technical Report, International Software Engineering Research Network.

Willis, M., Hiden, H., Hinchliffe, M., McKay, B. & Barton, G. (1997), 'Systems modelling using genetic programming', *Computers and Chemical Engineering* **21**, 1161–1166. Suplemento.

Wittig, G. (1995), Estimating Software Development Effort with Connectionist Models, Working Paper Series 33/95, Monash University.

Wolverton, R. W. (1974), 'The cost of developing large-scale software', *IEEE Transactions on Computers* **23**(6), 615–636.

Xu, X., Ester, M., Kriegel, H. P. & Sander, J. (1998), A distribution-based clustering algorithm for mining in large spatial databases, *in* 'Proceedings of the 14th ICDE', Orlando, Florida, pp. 324–331.

Xu, Z. & Khoshgoftaar, T. (2004), 'Identification of fuzzy models of software cost estimation', *Fuzzy Sets and Systems* **145**(1), 141–163.

Zadeh, L. A. (1965), 'Fuzzy set', *Information and Control* **8**, 338–353.

Zhang, P. (1993), 'Model selection via multifold cross validation', *Annals of Statistics* (21), 299–313.